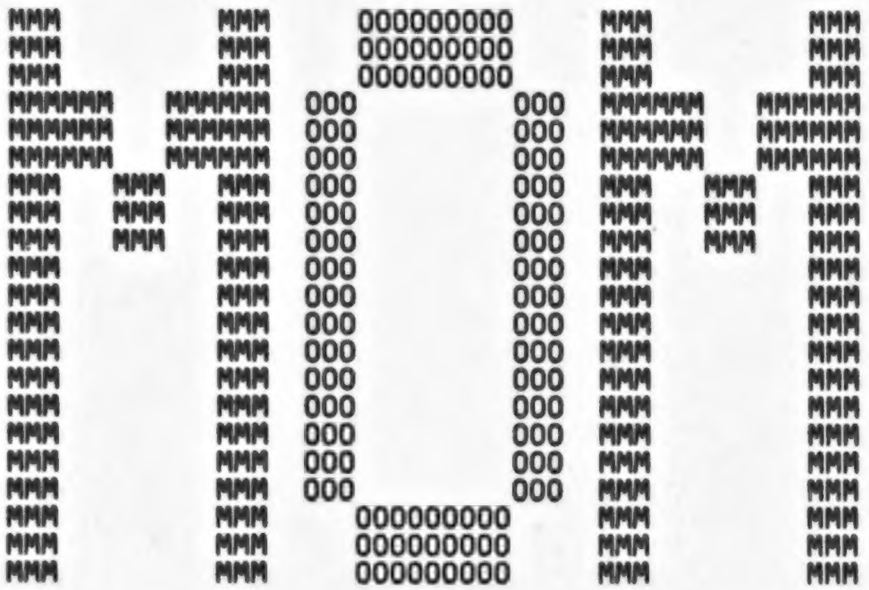
MMM MMM 000000000 MMM MMM
MMM MMM 000000000 MMM MMM
MMM MMM 000000000 MMM MMM
MMMMMMM MMMMMMM 000 000 MMMMMMM MMMMMM
MMMMMMM MMMMMMM 000 000 MMMMMMM MMMMMM
MMMMMMM MMMMMMM 000 000 MMMMMMM MMMMMM
MMM MMM MMM 000 000 MMM MMM MMM
MMM MMM MMM 000 000 MMM MMM MMM
MMM MMM MMM 000 000 MMM MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000000000 MMM MMM
MMM MMM 000000000 MMM MMM
MMM MMM 000000000 MMM MMM

FILEID**MOMLOAD

```
MM      MM    000000    MM      MM  LL            000000      AAAAAA    DDDDDDDD
MM      MM    000000    MM      MM  LL            000000      AAAAAA    DDDDDDDD
MMMM  MMMM  00      00  MMMM  MMMM  LL          00      00  AA      AA  DD      DD
MMMM  MMMM  00      00  MMMM  MMMM  LL          00      00  AA      AA  DD      DD
MM  MM  MM  00      00  MM  MM  MM  LL          00      00  AA      AA  DD      DD
MM  MM  MM  00      00  MM  MM  MM  LL          00      00  AA      AA  DD      DD
MM      MM  00      00  MM      MM  LL          00      00  AA      AA  DD      DD
MM      MM  00      00  MM      MM  LL          00      00  AA      AA  DD      DD
MM      MM  00      00  MM      MM  LL          00      00  AAAAAAAAAA  DD      DD
MM      MM  00      00  MM      MM  LL          00      00  AAAAAAAAAA  DD      DD
MM      MM  00      00  MM      MM  LL          00      00  AA      AA  DD      DD      ....
MM      MM  00      00  MM      MM  LL          00      00  AA      AA  DD      DD      ....
MM      MM    000000    MM      MM  LLLLLLLLLL    000000    AA      AA  DDDDDDDD      ....
MM      MM    000000    MM      MM  LLLLLLLLLL    000000    AA      AA  DDDDDDDD      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
;     1          0001  0 %TITLE 'Network Management Down Line Load Routines'
;     2          0002  0 MODULE MOMLOAD (
;     3          0003  0                 LANGUAGE (BLISS32),
;     4          0004  0                 ADDRESSING_MODE (NONEXTERNAL=LONG_RELATIVE),
;     5          0005  0                 ADDRESSING_MODE (EXTERNAL=LONG_RELATIVE),
;     6          0006  0                 IDENT = 'V04-000'
;     7          0007  0                 ) =
;     8          0008  1 BEGIN
;     9          0009  1 !
;    10          0010  1 !*****************************************************************************
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    13          0013  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    14          0014  1 !*  ALL RIGHTS RESERVED.                                                      *
;    15          0015  1 !*                                                                           *
;    16          0016  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    17          0017  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    18          0018  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    19          0019  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    20          0020  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    21          0021  1 !*  TRANSFERRED.                                                              *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    24          0024  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    25          0025  1 !*  CORPORATION.                                                              *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    28          0028  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    29          0029  1 !*                                                                           *
;    30          0030  1 !*                                                                           *
;    31          0031  1 !*****************************************************************************
;    32          0032  1 !
;    33          0033  1
;    34          0034  1 !++
;    35          0035  1 ! FACILITY:  DECnet-VAX V2.0 Maintenance Operations Module
;    36          0036  1 !
;    37          0037  1 ! ABSTRACT:
;    38          0038  1 !       This module contains routines to handle load maintenance operations,
;    39          0039  1 !       both target requested (automatic) and operator requested.
;    40          0040  1 !
;    41          0041  1 ! ENVIRONMENT:  VAX/VMS Operating System
;    42          0042  1 !
;    43          0043  1 ! AUTHOR:  Kathy Perko
;    44          0044  1 !
;    45          0045  1 ! CREATION DATE:  18-Feb-1983
;    46          0046  1 !
;    47          0047  1 ! MODIFIED BY:
;    48          0048  1 !       V03-004 MKP0004         Kathy Perko             2-May-1984
;    49          0049  1 !               Fix the LOAD trigger to use one channel for non-NI loads.
;    50          0050  1 !
;    51          0051  1 !       V03-003 MKP0003         Kathy Perko             11-Feb-1984
;    52          0052  1 !               Fix TRIGGER so it is sent on NI using Remote Console protocol
;    53          0053  1 !               instead of load/dump protocol.
;    54          0054  1 !
;    55          0055  1 !       V03-002 MKP0002         Kathy Perko             31-May-1983
;    56          0056  1 !               Fix load file open so that, if the target requests diagnostics,
;    57          0057  1 !               the diagnostics are loaded.
```

```
;    58          0058  1 !
;    59          0059  1 !       V03-001 MKP0001         Kathy Perko             10-May-1983
;    60          0060  1 !               Fix check on load number requested by target so it wraps
;    61          0061  1 !               from FF to 00.  Only trigger target if loading the
;    62          0062  1 !               secondary loader first.
;    63          0063  1 !
;    64          0064  1 !--
;    65          0065  1
```

```
;    67          0066  1 %SBTTL 'Declarations'
;    68          0067  1
;    69          0068  1 !
;    70          0069  1 ! TABLE OF CONTENTS:
;    71          0070  1 !
;    72          0071  1
;    73          0072  1 FORWARD ROUTINE
;    74          0073  1     mom$load                : NOVALUE,
;    75          0074  1     mom_load_trigger,
;    76          0075  1     mom_mblkload,
;    77          0076  1     mom_load_sys_file,
;    78          0077  1     mom_load_cc_file,
;    79          0078  1     mom_secload,
;    80          0079  1     mom_xmit_load_frame,
;    81          0080  1     mom_openloadfile,
;    82          0081  1     mom_readloadfile    : NOVALUE,
;    83          0082  1     mom_check_label_blk : NOVALUE,
;    84          0083  1     mom$loadhandler;
;    85          0084  1
;    86          0085  1 !
;    87          0086  1 ! INCLUDE FILES:
;    88          0087  1 !
;    89          0088  1
;    90          0089  1 LIBRARY 'LIB$:MOMLIB.L32';
;    91          0090  1 LIBRARY 'SHRLIB$:NMALIBRY.L32';
;    92          0091  1 LIBRARY 'SHRLIB$:EVCDEF.L32';
;    93          0092  1 LIBRARY 'SHRLIB$:NET.L32';
;    94          0093  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    95          0094  1
;    96          0095  1 !
;    97          0096  1 ! MOM$K_LOADBUFSIZ must be large enough to accomodate the entire secondary
;    98          0097  1 ! load image, since the secondary loader is always sent in one transmit.
;    99          0098  1 ! MOM$K_SEGBLKCNT is used to determine the number of 32 word blocks in
;   100          0099  1 ! each MOP transmit for a multiblock load (tertiary load and operating
;   101          0100  1 ! system load).
;   102          0101  1 !
;   103          0102  1 LITERAL
;   104          0103  1     mom$k_loadbufsiz = 1536,
;   105          0104  1     mom$k_segblkcnt = 4;        ! Number of 32-word blocks in a multiblock
;   106          0105  1                                 !       load segment
;   107          0106  1
;   108          0107  1 !
;   109          0108  1 ! OWN STORAGE:
;   110          0109  1 !
;   111          0110  1
;   112          0111  1 OWN
;   113          0112  1     mom$l_baseadr,                  ! Base address of load segment
;   114          0113  1     mom$l_blkcnt,                   ! Number of blocks in buffer
;   115          0114  1     mom$l_loadsize,                 ! Size of image in 32-word blocks
;   116          0115  1     mom$l_transfer,                 ! Image transfer address
;   117          0116  1     mom$w_pgmdetail : WORD,         ! Program error detail
;   118          0117  1     mom$w_first_load_frame;         ! Indicates if first load frame for
;   119          0118  1                                     ! a multiblock load has been sent.
;   120          0119  1 !
;   121          0120  1 ! The following buffers are used for downline loading.
;   122          0121  1 !
;   123          0122  1 ! MOM$T_LOADBUFFER is used for transmitting memory image data.  There
```

```
;  124     0123  1 ! are 6 bytes of overhead at the beginning of the buffer to hold MOP
;  125     0124  1 ! message information.  There are 4 bytes of overhead at the end of
;  126     0125  1 ! the buffer to contain the transfer address if it is needed and the
;  127     0126  1 ! image data takes up the entire buffer.  MOM$T_READBUFFER is the
;  128     0127  1 ! center of MOM$T_LOADBUFFER.  The image data is read from disk a block
;  129     0128  1 ! at a time, and transmitted piece by piece directly
;  130     0129  1 ! from this buffer which is why the overhead bytes are required.
;  131     0130  1 ! The MOM$Q_DATADSC is used to describe the extent of the image data
;  132     0131  1 ! read in to MOM$T_READBUFFER.
;  133     0132  1 !
;  134     0133  1 LITERAL
;  135     0134  1     mom$k_maxsecsiz = 1498 - 6 - 4;
;  136     0135  1 OWN
;  137     0136  1     mom$t_cc_wrap_buf: BBLOCK [mom$k_loadbufsiz],
;  138     0137  1     mom$t_loadbuffer : BBLOCK [6 + mom$k_loadbufsiz + 4];
;  139     0138  1
;  140     0139  1 BIND
;  141     0140  1     mom$t_readbuffer = mom$t_loadbuffer + 6
;  142     0141  1                      : BBLOCK [mom$k_loadbufsiz],
;  143     0142  1     mom$q_loadbfdsc  = UPLIT (6 + mom$k_loadbufsiz + 4, mom$t_loadbuffer)
;  144     0143  1                      : VECTOR [2],
;  145     0144  1     mom$q_readbfdsc  = UPLIT (mom$k_loadbufsiz, mom$t_readbuffer)
;  146     0145  1                      : VECTOR [2];
;  147     0146  1 OWN
;  148     0147  1     mom$q_datadsc    : VECTOR [2]
;  149     0148  1                        INITIAL (0, mom$t_readbuffer);
;  150     0149  1
;  151     0150  1 !
;  152     0151  1 ! EXTERNAL REFERENCES:
;  153     0152  1 !
;  154     0153  1
;  155     0154  1 $mom_externals;                          ! Macro to define common externals
;  156     0155  1
;  157     0156  1 EXTERNAL LITERAL
;  158     0157  1     mom$_unsmopdev,
;  159     0158  1     mom$_imgrecsiz,
;  160     0159  1     mom$_invccfil,
;  161     0160  1     mdt$gk_mopdevcnt;
;  162     0161  1
;  163     0162  1 EXTERNAL
;  164     0163  1     mom$ab_mopdevices : BBLOCKVECTOR [0,mdt$k_entrylen],
;  165     0164  1     mom$gq_timeout    : VECTOR [0],
;  166     0165  1     mom$npa_mopload;
;  167     0166  1
;  168     0167  1 EXTERNAL ROUTINE
;  169     0168  1     nma$nparse,
;  170     0169  1     mom$bld_reply,
;  171     0170  1     mom$bldmopboot,
;  172     0171  1     mom$bldmopplt,
;  173     0172  1     mom$chk_mop_error,
;  174     0173  1     mom$debug_txt,
;  175     0174  1     mom$error,
;  176     0175  1     mom$init_cib,
;  177     0176  1     mom$log_event,
;  178     0177  1     mom$mopopen,
;  179     0178  1     mom$mopsndrcv,
;  180     0179  1     mom$mopsetsubstate,
```

```
;  181          0180  1      mom$srvclose,
;  182          0181  1      mom$srvopen,
;  183          0182  1      mom$srvread,
;  184          0183  1      mom$srvrewind,
;  185          0184  1      mom$srvwrite;
;  186          0185  1
;  187          0186  1
```

```
;  189          0187  1 %SBTTL 'mom$load  Perform a downline system load'
;  190          0188  1 GLOBAL ROUTINE mom$load : NOVALUE =
;  191          0189  1
;  192          0190  1 !++
;  193          0191  1 ! FUNCTIONAL DESCRIPTION:
;  194          0192  1 !       This routine performs the downline system load function.
;  195          0193  1 !
;  196          0194  1 ! INPUTS:
;  197          0195  1 !       CIB - Channel Information Block for MOP QIO channel to the circuit
;  198          0196  1 !             over which to do the down line load.
;  199          0197  1 !
;  200          0198  1 ! ROUTINE VALUE:
;  201          0199  1 ! COMPLETION CODES:
;  202          0200  1 !
;  203          0201  1 !       Signal errors.
;  204          0202  1 !
;  205          0203  1 !--
;  206          0204  1
;  207          0205  2 BEGIN
;  208          0206  2
;  209          0207  2 LOCAL
;  210          0208  2     fldadr,
;  211          0209  2     fldsize,
;  212          0210  2     loadflag,
;  213          0211  2     msgdsc : VECTOR [2],
;  214          0212  2     msgsize,
;  215          0213  2     snddsc : VECTOR [2],
;  216          0214  2     status;
;  217          0215  2 !
;  218          0216  2 ! Enable condition handler to perform cleanup after load function.
;  219          0217  2 !
;  220          0218  2 ENABLE mom$loadhandler;
;  221          0219  2 !
;  222          0220  2 ! Set the circuit substate.
;  223          0221  2 !
;  224          0222  2 IF .mom$gl_service_flags [mom$v_autoservice] THEN
;  225          0223  2     mom$mopsetsubstate (nma$c_linss_alo,
;  226          0224  2                         .mom$ab_cib [cib$l_chan])  ! -AUTOLOADING
;  227          0225  2 ELSE
;  228          0226  2 !
;  229          0227  2 ! If doing an operator requested load, set the circuit substate to
;  230          0228  2 ! loading and trigger the target's load device.  If it doesn't respond
;  231          0229  2 ! to the trigger, mom_load_trigger will not return here.
;  232          0230  2 !
;  233          0231  3     BEGIN
;  234          0232  3     !
;  235          0233  3     ! Open the I/O channel over which to do the load.
;  236          0234  3     !
;  237          0235  3     mom$mopopen (mom$ab_cib [cib$l_chan]);
;  238          0236  3     mom$mopsetsubstate (nma$c_linss_loa,
;  239          0237  3                         .mom$ab_cib [cib$l_chan]); ! -LOADING
;  240          0238  3     !
;  241          0239  3     ! Set up Channel Information Block for the load channel.  For NI circuits,
;  242          0240  3     ! this sets up the NI protocol (in this case load/dump) the circuit, and
;  243          0241  3     ! associates it with a specific NI destination.
;  244          0242  3     !
;  245          0243  3     mom$init_CIB (mom$ab_cib,           ! Channel Information Block addr
```

```
;   246     0244  3                     nma$c_fnc_loa,          ! Function = load
;   247     0245  3                     svd$gk_pcno_pha,        ! NI Phycial Address
;   248     0246  3                     svd$gk_pcno_add,        ! Node address
;   249     0247  3                     svd$gk_pcno_hwa);       ! NI hardware address
;   250     0248  2         IF .mom$ab_service_data [svd$gk_pcno_sty, svd$l_param] EQL nma$c_soft_secl
;   251     0249  2         THEN
;   252     0250  2             mom_load_trigger ();
;   253     0251  2         END;
;   254     0252  2     !
;   255     0253  2     ! Perform the load.
;   256     0254  2     !
;   257     0255  2     loadflag = true;                        ! Set the load retry flag
;   258     0256  2
;   259     0257  2     WHILE 1 DO
;   260     0258  2         BEGIN
;   261     0259  3         !
;   262     0260  3         ! Open the file to be loaded.
;   263     0261  3         !
;   264     0262  3         status = mom_openloadfile ();
;   265     0263  3         !
;   266     0264  3         ! Load it.
;   267     0265  3         !
;   268     0266  3         IF .status THEN
;   269     0267  4             BEGIN
;   270     0268  4             !
;   271     0269  4             ! Log event for load requested.  Responding to a multicast load request
;   272     0270  4             ! from the target, however, don't log the event.  It's not reasonable
;   273     0271  4             ! for every node on the target's NI to log events.  Only the host that
;   274     0272  4             ! performs the load will log them.
;   275     0273  4             !
;   276     0274  4             mom$gw_evt_code = evc$c_nma_als;        ! Event code (automatic service)
;   277     0275  4             mom$gb_evt_pser = evc$c_nma_pser_loa;
;   278     0276  4             mom$log_event (0,0);
;   279     0277  4
;   280     0278  4             !
;   281     0279  4             ! Output the trace message.
;   282     0280  4             !
;   283     0281  4             mom$debug_txt (dbg$c_srvtrc,
;   284     0282  5                 (SELECTONEU .mom$ab_service_data [svd$gk_pcno_sty, svd$l_param] OF
;   285     0283  5                     SET
;   286     0284  5                     [nma$c_soft_secl]: $ASCID ('Loading secondary bootstrap.');
;   287     0285  5                     [nma$c_soft_terl]: $ASCID ('Loading tertiary bootstrap.');
;   288     0286  5                     [nma$c_soft_osys]: $ASCID ('Loading operating system.');
;   289     0287  5                     TES)
;   290     0288  4                 );
;   291     0289  4
;   292     0290  4             SELECTONEU .mom$ab_service_data [svd$gk_pcno_sty, svd$l_param] OF
;   293     0291  4                 SET
;   294     0292  4
;   295     0293  4                 [nma$c_soft_secl]:
;   296     0294  4                     status = mom_secload (loadflag, msgdsc);
;   297     0295  4
;   298     0296  4                 [OTHERWISE]:
;   299     0297  4                     status = mom_mblkload (loadflag, msgdsc);
;   300     0298  4
;   301     0299  4                 TES;
;   302     0300  4     !                                                                              ;
```

```
;   303     0301  4          ! Close the load file.
;   304     0302  4          !
;   305     0303  4          mom$srvclose ();
;   306     0304  4
;   307     0305  4          IF .status THEN
;   308     0306  5              BEGIN
;   309     0307  5              !
;   310     0308  5              ! Log the file that was loaded.
;   311     0309  5              !
;   312     0310  5              mom$debug_txt ( dbg$c_srvtrc,
;   313     0311  6                  (SELECTONEU .mom$ab_service_data [svd$gk_pcno_sty, svd$l_param] OF
;   314     0312  6                      SET
;   315     0313  6                      [nma$c_soft_osys]: $ASCID ('Operating system loaded.');
;   316     0314  6                      [nma$c_soft_terl]: $ASCID ('Tertiary bootstrap loaded.');
;   317     0315  6                      [nma$c_soft_secl]: $ASCID ('Secondary bootstrap loaded.');
;   318     0316  6                      TES)
;   319     0317  5                  );
;   320     0318  5              !
;   321     0319  5              ! The load is complete if the operating system has been loaded.
;   322     0320  5              !
;   323     0321  5              IF .mom$ab_service_data [svd$gk_pcno_sty, svd$l_param]
;   324     0322  5                                                  EQLU nma$c_soft_osys THEN
;   325     0323  6                  BEGIN
;   326     0324  6                  mom$ab_msgblock [msb$l_flags] = 0;
;   327     0325  6                  mom$ab_msgblock [msb$b_code]  = nma$c_sts_suc;
;   328     0326  6                  EXITLOOP;
;   329     0327  5                  END;
;   330     0328  5              !
;   331     0329  5              ! Log "load successful" event for the secondary or tertiary loader.
;   332     0330  5              !
;   333     0331  5              mom$gb_evt_pser = evc$c_nma_pser_loa;
;   334     0332  5              mom$log_event (4,UPLIT (BYTE (nma$c_sts_suc, %X'ff', %X'ff', 0)));
;   335     0333  5              !
;   336     0334  5              ! Parse the received MOP message to get information about the
;   337     0335  5              ! next load attempt.
;   338     0336  5              !
;   339     0337  5              mom$ab_nparse_blk [npa$l_msgcnt] = .msgdsc [0];
;   340     0338  5              mom$ab_nparse_blk [npa$l_msgptr] = .msgdsc [1];
;   341     0339  5
;   342     0340  5              status = nma$nparse (mom$ab_nparse_blk, mom$npa_moploa);
;   343     0341  5
;   344     0342  4              END;
;   345     0343  3          END;
;   346     0344  3          !
;   347     0345  3          ! If the load failed on the first message then there are two cases:
;   348     0346  3          !   The load was an NI multicast load request, and some other host responded
;   349     0347  3          !       to the target first.  So, give up now.
;   350     0348  3          !   Otherwise, trigger the target's bootstrap and try the load again.
;   351     0349  3          !
;   352     0350  3          IF NOT .status THEN
;   353     0351  4              BEGIN
;   354     0352  4              IF .mom$gl_service_flags [mom$v_ni_multicast] THEN
;   355     0353  4                  EXITLOOP;
;   356     0354  4              IF .loadflag THEN
;   357     0355  5                  BEGIN
;   358     0356  5                  loadflag = false;    ! No more retries
;   359     0357  5                  mom_load_trigger ();
```

```
;   360          0358  5                END
;   361          0359  4            ELSE
;   362          0360  4                EXITLOOP;
;   363          0361  4
;   364          0362  4            END
;   365          0363  3        ELSE
;   366          0364  3            loadflag = FALSE;
;   367          0365  3
;   368          0366  3        END;
;   369          0367  2    !
;   370          0368  2    ! Return status.
;   371          0369  2    !
;   372          0370  2    mom$bld_reply (mom$ab_msgblock, msgsize);
;   373          0371  2    $signal_msg (mom$ab_nice_xmit_buf, .msgsize);
;   374          0372  2
;   375          0373  1 END;                                ! End of mom$load


;                                                           .TITLE  MOMLOAD Network Management Down Line Load Routi
                                                                    nes
                                                            .IDENT  \V04-000\

                                                            .PSECT  $PLIT$,NOWRT,NOEXE,2

                                                0000060A  00000 P.AAA:  .LONG   1546
                                                00000000'  00004        .ADDRESS MOM$T_LOADBUFFER
                                                00000600  00008 P.AAB:  .LONG   1536
                                                00000000'  0000C        .ADDRESS MOM$T_READBUFFER
61 64 6E 6F 63 65 73 20 67 6E 69 64 61 6F 4C  00010 P.AAD:  .ASCII  \Loading secondary bootstrap.\
      2E 70 61 72 74 73 74 6F 6F 62 20 79 72  0001F
                                                0000001C  0002C P.AAC:  .LONG   28
                                                00000000'  00030        .ADDRESS P.AAD
72 61 69 74 72 65 74 20 67 6E 69 64 61 6F 4C  00034 P.AAF:  .ASCII  \Loading tertiary bootstrap.\
      2E 70 61 72 74 73 74 6F 6F 62 20 79       00043
                                                0004F           .BLKB   1
                                                0000001B  00050 P.AAE:  .LONG   27
                                                00000000'  00054        .ADDRESS P.AAF
69 74 61 72 65 70 6F 20 67 6E 69 64 61 6F 4C  00058 P.AAH:  .ASCII  \Loading operating system.\
            2E 6D 65 74 73 79 73 20 67 6E      00067
                                                00071           .BLKB   3
                                                00000019  00074 P.AAG:  .LONG   25
                                                00000000'  00078        .ADDRESS P.AAH
65 74 73 79 73 20 67 6E 69 74 61 72 65 70 4F  0007C P.AAJ:  .ASCII  \Operating system loaded.\
                  2E 64 65 64 61 6F 6C 20 6D  0008B
                                                00000018  00094 P.AAI:  .LONG   24
                                                00000000'  00098        .ADDRESS P.AAJ
74 73 74 6F 6F 62 20 79 72 61 69 74 72 65 54  0009C P.AAL:  .ASCII  \Tertiary bootstrap loaded.\
      2E 64 65 64 61 6F 6C 20 70 61 72          000AB
                                                000B6           .BLKB   2
                                                0000001A  000B8 P.AAK:  .LONG   26
                                                00000000'  000BC        .ADDRESS P.AAL
73 74 6F 6F 62 20 79 72 61 64 6E 6F 63 65 53  000C0 P.AAN:  .ASCII  \Secondary bootstrap loaded.\
      2E 64 65 64 61 6F 6C 20 70 61 72 74       000CF
                                                000DB           .BLKB   1
                                                0000001B  000DC P.AAM:  .LONG   27
                                                00000000'  000E0        .ADDRESS P.AAN
                                          00 FF FF 01  000E4 P.AAO:  .BYTE   1, -1, -1, 0
```

```
                                                    B 14
                                                                .PSECT  $OWN$,NOEXE,2

                                                    00000 MOM$L_BASEADR:
                                                                .BLKB   4
                                                    00004 MOM$L_BLKCNT:
                                                                .BLKB   4
                                                    00008 MOM$L_LOADSIZE:
                                                                .BLKB   4
                                                    0000C MOM$L_TRANSFER:
                                                                .BLKB   4
                                                    00010 MOM$W_PGMDETAIL:
                                                                .BLKB   2
                                                    00012       .BLKB   2
                                                    00014 MOM$W_FIRST_LOAD_FRAME:
                                                                .BLKB   4
                                                    00018 MOM$T_CC_WRAP_BUF:
                                                                .BLKB   1536
                                                    00618 MOM$T_LOADBUFFER:
                                                                .BLKB   1546
                                                    00C22       .BLKB   2
                                         00000000   00C24 MOM$Q_DATADSC:
                                                                .LONG   0
                                         00000000'  00C28       .ADDRESS MOM$T_READBUFFER

                                                    MOM$T_READBUFFER=   MOM$T_LOADBUFFER+6
                                                    MOM$Q_LOADBFDSC=    P.AAA
                                                    MOM$Q_READBFDSC=    P.AAB
                                                                .EXTRN  MOM$GL_LOGMASK, MOM$GL_SVD_INDEX
                                                                .EXTRN  MOM$AB_SERVICE_DATA
                                                                .EXTRN  MOM$GB_FUNCTION
                                                                .EXTRN  MOM$GB_OPTION_BYTE
                                                                .EXTRN  MOM$GB_ENTITY_CODE
                                                                .EXTRN  MOM$AB_ENTITY_BUF
                                                                .EXTRN  MOM$GQ_ENTITY_BUF_DSC
                                                                .EXTRN  MOM$GL_SERVICE_FLAGS
                                                                .EXTRN  MOM$AB_NPARSE_BLK
                                                                .EXTRN  MOM$AB_NICE_RCV_BUF
                                                                .EXTRN  MOM$AB_NICE_XMIT_BUF
                                                                .EXTRN  MOM$GQ_NICE_RCV_BUF_DSC
                                                                .EXTRN  MOM$GL_NICE_RCV_MSG_LEN
                                                                .EXTRN  MOM$GQ_NICE_XMIT_BUF_DSC
                                                                .EXTRN  MOM$AB_MSGBLOCK
                                                                .EXTRN  MOM$AB_ACPQIO_BUFFER
                                                                .EXTRN  MOM$GQ_ACPQIO_BUF_DSC
                                                                .EXTRN  MOM$AB_CIB, MOM$AB_LOOP_CIB
                                                                .EXTRN  MOM$AB_TRIGGER_CIB
                                                                .EXTRN  MOM$AB_MOP_XMIT_BUF
                                                                .EXTRN  MOM$GQ_MOP_XMIT_BUF_DSC
                                                                .EXTRN  MOM$AB_MOP_RCV_BUF
                                                                .EXTRN  MOM$GQ_MOP_RCV_BUF_DSC
                                                                .EXTRN  MOM$AB_MOP_MSG, MOM$GQ_MOP_MSG_DSC
                                                                .EXTRN  MOM$GW_EVT_CODE
                                                                .EXTRN  MOM$GB_EVT_POPR
                                                                .EXTRN  MOM$GB_EVT_PRSN
                                                                .EXTRN  MOM$GB_EVT_PSER
                                                                .EXTRN  SVD$GK_PCNO_ADD
```

```
                                        .EXTRN  SVD$GK_PCNO_SDV
                                        .EXTRN  SVD$GK_PCNO_CPU
                                        .EXTRN  SVD$GK_PCNO_STY
                                        .EXTRN  SVD$GK_PCNO_DAD
                                        .EXTRN  SVD$GK_PCNO_DCT
                                        .EXTRN  SVD$GK_PCNO_IHO
                                        .EXTRN  SVD$GK_PCNO_NNA
                                        .EXTRN  SVD$GK_PCNO_SLI
                                        .EXTRN  SVD$GK_PCNO_SPA
                                        .EXTRN  SVD$GK_PCNO_HWA
                                        .EXTRN  SVD$GK_PCNO_SNV
                                        .EXTRN  SVD$GK_PCNO_LOA
                                        .EXTRN  SVD$GK_PCNO_SLO
                                        .EXTRN  SVD$GK_PCNO_TLO
                                        .EXTRN  SVD$GK_PCNO_DFL
                                        .EXTRN  SVD$GK_PCNO_SID
                                        .EXTRN  SVD$GK_PCNO_DUM
                                        .EXTRN  SVD$GK_PCNO_SDU
                                        .EXTRN  SVD$GK_PCNO_$HNA
                                        .EXTRN  SVD$GK_PCNO_$HHW
                                        .EXTRN  SVD$GK_PCNO_$FTY
                                        .EXTRN  SVD$GK_PCNO_PHA
                                        .EXTRN  SVD$GK_PCNO_$DA
                                        .EXTRN  SVD$GK_PCNO_LPC
                                        .EXTRN  SVD$GK_PCNO_LPL
                                        .EXTRN  SVD$GK_PCNO_LPD
                                        .EXTRN  SVD$GK_PCNO_LPH
                                        .EXTRN  SVD$GK_PCNO_LPA
                                        .EXTRN  SVD$GK_PCNO_LPN
                                        .EXTRN  SVD$GK_PCNO_$LNA
                                        .EXTRN  SVD$GK_PCNO_$LNH
                                        .EXTRN  SVD$GK_PCNO_LAN
                                        .EXTRN  SVD$GK_PCNO_$LNN
                                        .EXTRN  SVD$GK_PCNO_$LAH
                                        .EXTRN  SVD$GK_PCLI_STI
                                        .EXTRN  SVD$C_ENTRY_COUNT
                                        .EXTRN  MOM$_ONSMOPDEV, MOM$_IMGRECSIZ
                                        .EXTRN  MOM$_INVCCFIL, MDT$GR_MOPDEVCNT
                                        .EXTRN  MOM$AB_MOPDEVICES
                                        .EXTRN  MOM$GQ_TIMEOUT, MOM$NPA_MOPLOAD
                                        .EXTRN  NMA$NPARSE, MOM$BLD_REPLY
                                        .EXTRN  MOM$BLDMOPBOOT, MOM$BLDMOPPLT
                                        .EXTRN  MOM$CHK_MOP_ERROR
                                        .EXTRN  MOM$DEBUG_TXT, MOM$ERROR
                                        .EXTRN  MOM$INIT_CIB, MOM$LOG_EVENT
                                        .EXTRN  MOM$MOPOPEN, MOM$MOPSNDRCV
                                        .EXTRN  MOM$MOPSETSUBSTATE
                                        .EXTRN  MOM$SRVCLOSE, MOM$SRVOPEN
                                        .EXTRN  MOM$SRVREAD, MOM$SRVREWIND
                                        .EXTRN  MOM$SRVWRITE

                                        .PSECT  $CODE$,NOWRT,2

                        0FFC 00000      .ENTRY  MOM$LOAD, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0188
                                                R11
5B 00000000G   EF   9E 00002            MOVAB   MOM$GL_SERVICE_FLAGS, R11                     ;
5A 00000000G   EF   9E 00009            MOVAB   MOM$DEBUG_TXT, R10                            ;
```

```
                     59 00000000G   EF   9E 00010          MOVAB   MOM$LOG_EVENT, R9                                        ;
                     58 00000000G   EF   9E 00017          MOVAB   MOM$GB_EVT_PSER, R8                                     ;
                     57 00000000V   EF   9E 0001E          MOVAB   MOM_LOAD_TRIGGER, R7                                    ;
                     56 00000000G   EF   9E 00025          MOVAB   MOM$MOPSETSUBSTATE, R6                                  ;
                     55 00000000G   EF   9E 0002C          MOVAB   MOM$AB_MSGBLOCK, R5                                     ;
                     54 00000000G   EF   9E 00033          MOVAB   MOM$AB_CIB, R4                                          ;
                     53 00000000'   EF   9E 0003A          MOVAB   P.AAC, R3                                               ;
                     5E             18   C2 00041          SUBL2   #24, SP                                                 ;
                     6D      016D   CF   DE 00044          MOVAL   22$, (FP)                                               ; 0205
                     09             6B   E9 00049          BLBC    MOM$GL_SERVICE_FLAGS, 1$                                ; 0222
                                    64   DD 0004C          PUSHL   MOM$AB_CIB                                              ; 0224
                                    07   DD 0004E          PUSHL   #7                                                      ; 0223
                     66             02   FB 00050          CALLS   #2, MOM$MOPSETSUBSTATE                                  ;
                                    38   11 00053          BRB     2$                                                      ;
                                    54   DD 00055 1$:      PUSHL   R4                                                      ; 0235
          00000000G  EF             01   FB 00057          CALLS   #1, MOM$MOPOPEN                                         ;
                                    64   DD 0005E          PUSHL   MOM$AB_CIB                                              ; 0237
                                    03   DD 00060          PUSHL   #3                                                      ; 0236
                     66             02   FB 00062          CALLS   #2, MOM$MOPSETSUBSTATE                                  ;
                        00000000G   8F   DD 00065          PUSHL   #SVD$GK_PCNO_HWA                                        ; 0243
                        00000000G   8F   DD 0006B          PUSHL   #SVD$GK_PCNO_ADD                                        ;
                        00000000G   8F   DD 00071          PUSHL   #SVD$GK_PCNO_PHA                                        ;
                                    0F   DD 00077          PUSHL   #15                                                     ;
                                    54   DD 00079          PUSHL   R4                                                      ;
          00000000G  EF             05   FB 0007B          CALLS   #5, MOM$INIT_CIB                                        ;
                        00000000*   EF   D5 00082          TSTL    <<MOM$AB_SERVICE_DATA+<SVD$GK_PCNO_STY*137>-            ; 0248
                                                                    >+9>                                                    ;
                                    03   12 00088          BNEQ    2$                                                      ;
                     67             00   FB 0008A          CALLS   #0, MOM_LOAD_TRIGGER                                    ; 0250
                     6E             01   D0 0008D 2$:      MOVL    #1, LOADFLAG                                            ; 0255
          00000000V  EF             00   FB 00090 3$:      CALLS   #0, MOM_OPENLOADFILE                                    ; 0262
                     52             50   D0 00097          MOVL    R0, STATUS                                              ;
                     6D             52   E9 0009A          BLBC    STATUS, 11$                                             ; 0266
          00000000G  EF             03   90 0009D          MOVB    #3, MOM$GW_EVT_CODE                                     ; 0274
                                    68   94 000A4          CLRB    MOM$GB_EVT_PSER                                         ; 0275
                                    7E   7C 000A6          CLRQ    -(SP)                                                   ; 0276
                     69             02   FB 000A8          CALLS   #2, MOM$LOG_EVENT                                       ;
                     50 00000000*   EF   D0 000AB          MOVL    <<MOM$AB_SERVICE_DATA+<SVD$GK_PCNO_STY*137>-            ; 0282
                                                                    >+9>, R0                                                ;
                                    05   12 000B2          BNEQ    4$                                                      ; 0284
                     51             63   9E 000B4          MOVAB   P.AAC, R1                                               ;
                                    09   11 000B7          BRB     5$                                                      ;
                     01             50   D1 000B9 4$:      CMPL    R0, #1                                                  ; 0285
                                    08   12 000BC          BNEQ    6$                                                      ;
                     51      24     A3   9E 000BE          MOVAB   P.AAE, R1                                               ;
                                    51   DD 000C2 5$:      PUSHL   R1                                                      ;
                                    10   11 000C4          BRB     8$                                                      ;
                     02             50   D1 000C6 6$:      CMPL    R0, #2                                                  ; 0286
                                    05   13 000C9          BEQL    7$                                                      ;
                     7E             01   CE 000CB          MNEGL   #1, -(SP)                                               ;
                                    06   11 000CE          BRB     8$                                                      ;
                     50      48     A3   9E 000D0 7$:      MOVAB   P.AAG, R0                                               ;
                                    50   DD 000D4          PUSHL   R0                                                      ;
                                    06   DD 000D6 8$:      PUSHL   #6                                                      ; 0281
                     6A             02   FB 000D8          CALLS   #2, MOM$DEBUG_TXT                                       ;
                     50 00000000*   EF   D0 000DB          MOVL    <<MOM$AB_SERVICE_DATA+<SVD$GK_PCNO_STY*137>-            ; 0290
                                                                    >+9>, R0                                                ;
```

```
                                 0F  12  000E2          BNEQ    9$                                                      ; 0293
                              10 AE  9F  000E4          PUSHAB  MSGDSC                                                  ; 0294
                              04 AE  9F  000E7          PUSHAB  LOADFLAG
                00000000V     EF     02  FB  000EA      CALLS   #2, MOM_SECLOAD
                                     0D  11  000F1      BRB     10$
                              10 AE  9F  000F3  9$:      PUSHAB  MSGDSC                                                  ; 0297
                              04 AE  9F  000F6          PUSHAB  LOADFLAG
                00000000V     EF     02  FB  000F9      CALLS   #2, MOM_MBLKLOAD
                52                   50  D0  00100  10$:  MOVL    R0, STATUS
                00000000G     EF     00  FB  00103      CALLS   #0, MOM$SRVCLOSE                                        ; 0303
                72                   52  E9  0010A  11$:  BLBC    STATUS, 18$                                           ; 0305
                50 00000000*  EF     D0  0010D          MOVL    <<MOM$AB_SERVICE_DATA+<SVD$GK_PCNO_STY*137>-          ; 0311
                                                                >+9>, R0
                02                   50  D1  00114      CMPL    R0, #2                                                ; 0313
                                     06  12  00117      BNEQ    12$
                51         68        A3  9E  00119      MOVAB   P.AAI, R1
                                     0A  11  0011D      BRB     13$
                01                   50  D1  0011F  12$:  CMPL    R0, #1                                                ; 0314
                                     09  12  00122      BNEQ    14$
                51         008C      C3  9E  00124      MOVAB   P.AAK, R1
                                     51  DD  00129  13$:  PUSHL   R1
                                     10  11  0012B      BRB     16$
                                     50  D5  0012D  14$:  TSTL    R0                                                    ; 0315
                                     05  13  0012F      BEQL    15$
                7E                   01  CE  00131      MNEGL   #1, -(SP)
                                     07  11  00134      BRB     16$
                50         00B0      C3  9E  00136  15$:  MOVAB   P.AAM, R0
                                     50  DD  0013B      PUSHL   R0
                                     06  DD  0013D  16$:  PUSHL   #6                                                    ; 0310
                6A                   02  FB  0013F      CALLS   #2, MOM$DEBUG_TXT
                02 00000000*  EF     D1  00142          CMPL    <<MOM$AB_SERVICE_DATA+<SVD$GK_PCNO_STY*137>-          ; 0322
                                                                >+9>, #2
                                     08  12  00149      BNEQ    17$
                                     65  D4  0014B      CLRL    MOM$AB_MSGBLOCK                                       ; 0324
                      04    A5       01  90  0014D      MOVB    #1, MOM$AB_MSGBLOCK+4                                 ; 0325
                                     3F  11  00151      BRB     21$                                                   ; 0323
                                     68  94  00153  17$:  CLRB    MOM$GB_EVT_PSER                                       ; 0331
                           00B8      C3  9F  00155      PUSHAB  P.AAO                                                 ; 0332
                                     04  DD  00159      PUSHL   #4
                69                   02  FB  0015B      CALLS   #2, MOM$LOG_EVENT
                00000000G     EF  10 AE  7D  0015E      MOVQ    MSGDSC, MOM$AB_NPARSE_BLK+4                           ; 0337
                           00000000G EF  9F  00166      PUSHAB  MOM$NPA_MOPLOAD                                       ; 0340
                           00000000G EF  9F  0016C      PUSHAB  MOM$AB_NPARSE_BLK
                00000000G     EF     02  FB  00172      CALLS   #2, NMA$NPARSE
                52                   50  D0  00179      MOVL    R0, STATUS
                0E                   52  E8  0017C      BLBS    STATUS, 19$                                           ; 0350
             0F 6B                   05  E0  0017F  18$:  BBS     #5, MOM$GL_SERVICE_FLAGS, 21$                         ; 0352
                0C                   6E  E9  00183      BLBC    LOADFLAG, 21$                                         ; 0354
                                     6E  D4  00186      CLRL    LOADFLAG                                              ; 0356
                67                   00  FB  00188      CALLS   #0, MOM_LOAD_TRIGGER                                  ; 0357
                                     02  11  0018B      BRB     20$                                                   ; 0354
                                     6E  D4  0018D  19$:  CLRL    LOADFLAG                                              ; 0364
                                FEFE 31  0018F  20$:  BRW     3$                                                    ; 0257
                           04   AE  9F  00192  21$:  PUSHAB  MSGSIZE                                                  ; 0370
                                     55  DD  00195      PUSHL   R5
                00000000G     EF     02  FB  00197      CALLS   #2, MOM$BLD_REPLY
                           04   AE  DD  0019E          PUSHL   MSGSIZE                                               ; 0371
```

```
                                00000000G  EF   9F  001A1          PUSHAB  MOM$AB_NICE_XMIT_BUF                                    ;
                                02070000   8F   DD  001A7          PUSHL   #34013T84                                               ;
                      00000000G 00         03   FB  001AD          CALLS   #3, LIB$SIGNAL                                          ;
                                           04   001B4              RET                                                             ; 0373
                                     0000  001B5  22$:              .WORD   Save nothing                                            ; 0205
                                     7E    D4   001B7              CLRL    -(SP)                                                   ;
                                     5E    DD   001B9              PUSHL   SP                                                      ;
                                7E   04    AC   7D  001BB          MOVQ    4(AP), -(SP)                                            ;
                      00000000V EF         03   FB  001BF          CALLS   #3, MOM$LOADHANDLER                                     ;
                                           04   001C6              RET                                                             ;

; Routine Size:  455 bytes,    Routine Base:  $CODE$ + 0000

;  376          0374  1
```

```
;  378     0375  1 %SBTTL 'mom_load_trigger  Trigger target node'
;  379     0376  1 ROUTINE mom_load_trigger =
;  380     0377  1
;  381     0378  1 !++
;  382     0379  1 ! FUNCTIONAL DESCRIPTION:
;  383     0380  1 !       This routine sends a boot message to the target system and
;  384     0381  1 !       parses the MOP message sent in response to the boot message.
;  385     0382  1 !       Two channels to the NI are used: one to send the boot message
;  386     0383  1 !       using the remote console protocol, and one to receive the response
;  387     0384  1 !       which will be sent using the load/dump protocol.
;  388     0385  1 !
;  389     0386  1 ! FORMAL PARAMETERS:
;  390     0387  1 !       NONE
;  391     0388  1 !
;  392     0389  1 ! ROUTINE VALUE:
;  393     0390  1 ! COMPLETION CODES:
;  394     0391  1 !
;  395     0392  1 !       Signal errors.
;  396     0393  1 !
;  397     0394  1 !--
;  398     0395  1
;  399     0396  2 BEGIN
;  400     0397  2
;  401     0398  2 LOCAL
;  402     0399  2     save_service_timer,
;  403     0400  2     xmit_CIB :  REF BBLOCK,
;  404     0401  2     rcv_CIB :   REF BBLOCK,
;  405     0402  2     snddsc :    VECTOR [2],
;  406     0403  2     msgdsc :    VECTOR [2],
;  407     0404  2     msgsize,
;  408     0405  2     status;
;  409     0406  2
;  410     0407  2 rcv_CIB = mom$ab_cib;
;  411     0408  2 xmit_CIB = mom$ab_cib;
;  412     0409  2 !
;  413     0410  2 ! Get a channel to the NI on which to send the boot message to the target.
;  414     0411  2 ! This channel is necessary because the boot message must be sent using the
;  415     0412  2 ! remote console NI protocol.  The response "load me" message from the target
;  416     0413  2 ! will be sent to the load/dump NI protocol.
;  417     0414  2 !
;  418     0415  2 IF .mom$gl_service_flags [mom$v_ni_circ] THEN
;  419     0416  2     BEGIN
;  420     0417  2     xmit_CIB = mom$ab_trigger_cib;
;  421     0418  2     mom$mopopen (xmit_CIB [cib$l_chan]);
;  422     0419  2     mom$init_CIB (.xmit_CIB,                    ! Channel Information Block addr
;  423     0420  2                   nma$c_fnc_tri,                ! Function = trigger
;  424     0421  2                   svd$gk_pcno_pha,              ! NI physical address of target
;  425     0422  2                   svd$gk_pcno_add,              ! Node address of target
;  426     0423  2                   svd$gk_pcno_hwa);             ! NI hardware address of target
;  427     0424  2     END;
;  428     0425  2 !
;  429     0426  2 ! Build the trigger (old 'enter MOP mode', new 'boot') message.
;  430     0427  2 !
;  431     0428  2 mom$bldmopboot (snddsc);
;  432     0429  2 mom$def_ _txt (dbg$c_srvtrc,
;  433     0430  2                 $ASCID ('Triggering remote bootstrap'));
;  434     0431  2 !
```

```
;   435          0432  2 ! Use an extra long timeout period because the PLUTO self test (which it
;   436          0433  2 ! goes through for every boot) takes a while.
;   437          0434  2 !
;   438          0435  2 xmit_CIB [cib$l_retry_cnt] = 2;
;   439          0436  2 save_service_timer = .mom$gq_timeout [0];
;   440          0437  2 mom$gq_timeout [0] = .mom$gq_timeout [0] * 10;
;   441          0438  2 msgdsc [1] = .mom$gq_mop_rcv_buf_dsc [1];
;   442          0439  2 !
;   443          0440  2 ! Send the boot message and listen for the target's response.  It should be
;   444          0441  2 ! a Program Load Request.
;   445          0442  2 !
;   446          0443  2 status = mom$mopsndrcv (.xmit_CIB, snddsc,
;   447          0444  2                         .rcv_CIB, mom$gq_mop_rcv_buf_dsc,
;   448          0445  2                         msgdsc [0],
;   449          0446  2                         0);             ! Don't skip program load requests
;   450          0447  2 mom$chk_mop_error (.status);
;   451          0448  2 mom$gq_timeout [0] = .save_service_timer;
;   452          0449  2
;   453          0450  2 !
;   454          0451  2 ! Parse the returned MOP message to make sure it's a valid Program Load
;   455          0452  2 ! Request.
;   456          0453  2 !
;   457          0454  2 mom$ab_nparse_blk [npa$l_msgcnt] = .msgdsc [0];
;   458          0455  2 mom$ab_nparse_blk [npa$l_msgptr] = .msgdsc [1];
;   459          0456  2 status = nma$nparse (mom$ab_nparse_blk, mom$npa_mopload);
;   460          0457  2 IF NOT .status THEN
;   461          0458  2     BEGIN
;   462          0459  2     mom$bld_reply (mom$ab_msgblock, msgsize);
;   463          0460  2     $signal_msg (mom$ab_nice_xmit_buf, .msgsize);
;   464          0461  2     END;
;   465          0462  2 !
;   466          0463  2 ! Deassign the MOP channel used to send the boot message.
;   467          0464  2 !
;   468          0465  2 IF .mom$gl_service_flags [mom$v_ni_circ] THEN
;   469          0466  2     $DASSGN (CHAN = .xmit_CIB [cib$l_chan]);
;   470          0467  2 RETURN .status;
;   471          0468  1 END;                                    ! End of mom_load_trigger


                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

6F  6D  65  72  20  67  6E  69  72  65  67  67  69  72  54  000E8 P.AAQ:  .ASCII  \Triggering remote bootstrap\                ;
            70  61  72  74  73  74  6F  6F  62  20  65  74  000F7                                                           ;
                                                        00103       .BLKB   1
                                                0000001B 00104 P.AAP:   .LONG   27                                      ;
                                                00000000' 00108         .ADDRESS P.AAQ                                  ;

                                                                .EXTRN  SYS$DASSGN

                                                                .PSECT  $CODE$,NOWRT,2

                                                        00FC 00000 MOM_LOAD_TRIGGER:
                                                                .WORD   Save R2,R3,R4,R5,R6,R7                          ; 0376
                                        57 00000000G    EF  9E 00002            MOVAB   MOM$AB_CIB, R7                  ;
                                        56 00000000G    EF  9E 00009            MOVAB   MOM$GL_SERVICE_FLAGS, R6        ;
                                        55 00000000G    EF  9E 00010            MOVAB   MOM$GQ_TIMEOUT, R5              ;
```

```
                     5E            14  C2 00017       SUBL2   #20, SP                                       ;
                     53            67  9E 0001A       MOVAB   MOM$AB_CIB, RCV_CIB                           ; 0407
                     52            67  9E 0001D       MOVAB   MOM$AB_CIB, XMIT_CIB                          ; 0408
       2D            66            01  E1 00020       BBC     #1, MOM$GL_SERVICE_FLAGS, 1$                  ; 0415
                     52 00000000G  EF  9E 00024       MOVAB   MOM$AB_TRIGGER_CIB, XMIT_CIB                  ; 0417
                                   52  DD 0002B       PUSHL   XMIT_CIB                                      ; 0418
          00000000G  EF            01  FB 0002D       CALLS   #1, MOM$MOPOPEN                               ;
                        00000000G  8F  DD 00034       PUSHL   #SVD$GK_PCNO_HWA                              ; 0419
                        00000000G  8F  DD 0003A       PUSHL   #SVD$GK_PCNO_ADD                              ;
                        00000000G  8F  DD 00040       PUSHL   #SVD$GK_PCNO_PHA                              ;
                                   11  DD 00046       PUSHL   #17                                           ;
                                   52  DD 00048       PUSHL   XMIT_CIB                                      ;
          00000000G  EF            05  FB 0004A       CALLS   #5, MOM$INIT_CIB                              ;
                               0C  AE  9F 00051 1$:   PUSHAB  SNDDSC                                        ; 0428
          00000000G  EF            01  FB 00054       CALLS   #1, MOM$BLDMOPBOOT                            ;
                        00000000'  EF  9F 0005B       PUSHAB  P.AAP                                         ; 0430
                                   06  DD 00061       PUSHL   #6                                            ; 0429
          00000000G  EF            02  FB 00063       CALLS   #2, MOM$DEBUG_TXT                             ;
                12   A2            02  D0 0006A       MOVL    #2, 18(XMIT_CIB)                              ; 0435
                     54            65  D0 0006E       MOVL    MOM$GQ_TIMEOUT, SAVE_SERVICE_TIMER            ; 0436
                     65            0A  C4 00071       MULL2   #10, MOM$GQ_TIMEOUT                           ; 0437
                08   AE 00000000G  EF  D0 00074       MOVL    MOM$GQ_MOP_RCV_BUF_DSC+4, MSGDSC+4            ; 0438
                                   7E  D4 0007C       CLRL    -(SP)                                         ; 0443
                               08  AE  9F 0007E       PUSHAB  MSGDSC                                        ; 0445
                        00000000G  EF  9F 00081       PUSHAB  MOM$GQ_MOP_RCV_BUF_DSC                        ; 0443
                                   53  DD 00087       PUSHL   RCV_CIB                                       ; 0444
                               1C  AE  9F 00089       PUSHAB  SNDDSC                                        ; 0443
                                   52  DD 0008C       PUSHL   XMIT_CIB                                      ;
          00000000G  EF            06  FB 0008E       CALLS   #6, MOM$MOPSNDRCV                             ;
                     53            50  D0 00095       MOVL    R0, STATUS                                    ;
                                   53  DD 00098       PUSHL   STATUS                                        ; 0447
          00000000G  EF            01  FB 0009A       CALLS   #1, MOM$CHK_MOP_ERROR                         ;
                     65            54  D0 000A1       MOVL    SAVE_SERVICE_TIMER, MOM$GQ_TIMEOUT            ; 0448
          00000000G  EF        04  AE  7D 000A4       MOVQ    MSGDSC, MOM$AB_NPARSE_BLK+4                   ; 0454
                        00000000G  EF  9F 000AC       PUSHAB  MOM$NPA_MOPLOAD                               ; 0456
                        00000000G  EF  9F 000B2       PUSHAB  MOM$AB_NPARSE_BLK                             ;
          00000000G  EF            02  FB 000B8       CALLS   #2, NMA$NPARSE                                ;
                     53            50  D0 000BF       MOVL    R0, STATUS                                    ;
                     24            53  E8 000C2       BLBS    STATUS, 2$                                    ; 0457
                                   5E  DD 000C5       PUSHL   SP                                            ; 0459
                        00000000G  EF  9F 000C7       PUSHAB  MOM$AB_MSGBLOCK                               ;
          00000000G  EF            02  FB 000CD       CALLS   #2, MOM$BLD_REPLY                             ;
                                   6E  DD 000D4       PUSHL   MSGSIZE                                       ; 0460
                        00000000G  EF  9F 000D6       PUSHAB  MOM$AB_NICE_XMIT_BUF                          ;
                        02070000   8F  DD 000DC       PUSHL   #34013184                                     ;
          00000000G  00            03  FB 000E2       CALLS   #3, LIB$SIGNAL                                ;
       09            66            01  E1 000E9 2$:   BBC     #1, MOM$GL_SERVICE_FLAGS, 3$                  ; 0465
                                   62  DD 000ED       PUSHL   (XMIT_CIB)                                    ; 0466
          00000000G  00            01  FB 000EF       CALLS   #1, SYS$DASSGN                                ;
                     50            53  D0 000F6 3$:   MOVL    STATUS, R0                                    ; 0467
                                       04 000F9       RET                                                   ; 0468

; Routine Size:  250 bytes,    Routine Base:  $CODE$ + 01C7
```

```
;  473          0469  1 %SBTTL 'mom_mblkload      Perform general multiblock load'
;  474          0470  1 ROUTINE mom_mblkload (loadflag, msgdsc) =
;  475          0471  1
;  476          0472  1 !++
;  477          0473  1 ! FUNCTIONAL DESCRIPTION:
;  478          0474  1 !
;  479          0475  1 !       This routine performs a general multiblock system load.  It is
;  480          0476  1 !       used to down-line load the tertiary loader and the operating
;  481          0477  1 !       system images.
;  482          0478  1 !
;  483          0479  1 ! FORMAL PARAMETERS:
;  484          0480  1 !
;  485          0481  1 !       LOADFLAG        Address of load retry flag (TRUE=>if load failed
;  486          0482  1 !                       it failed on the first message exchange).
;  487          0483  1 !       MSGDSC          Address of descriptor for received MOP message.
;  488          0484  1 !
;  489          0485  1 ! ROUTINE VALUE:
;  490          0486  1 ! COMPLETION CODES:
;  491          0487  1 !
;  492          0488  1 !       Signal errors.
;  493          0489  1 !
;  494          0490  1 !--
;  495          0491  1
;  496          0492  2 BEGIN
;  497          0493  2
;  498          0494  2 MAP
;  499          0495  2     msgdsc : REF VECTOR;
;  500          0496  2
;  501          0497  2 OWN
;  502          0498  2     PLT_response;
;  503          0499  2
;  504          0500  2 LOCAL
;  505          0501  2     status,
;  506          0502  2     snddsc: VECTOR [2],
;  507          0503  2     loadnum,
;  508          0504  2     skip_msg_dsc_addr;
;  509          0505  2
;  510          0506  2 BIND
;  511          0507  2     PLT_response_dsc = UPLIT (2, PLT_response);
;  512          0508  2
;  513          0509  2 msgdsc [1] = mom$ab_mop_rcv_buf;
;  514          0510  2
;  515          0511  2 !
;  516          0512  2 ! Send the load file to the target, a frame at a time, getting a response
;  517          0513  2 ! from the target for each frame.  If loading the console carrier code, the
;  518          0514  2 ! file format is different.
;  519          0515  2 !
;  520          0516  2 mom$w_first_load_frame = 1;
;  521          0517  2 IF .mom$gl_service_flags [mom$v_console_carrier_load] THEN
;  522          0518  2     status = mom_load_cc_file (.loadflag, .msgdsc, loadnum)
;  523          0519  2 ELSE
;  524          0520  2     status = mom_load_sys_file (.loadflag, .msgdsc, loadnum);
;  525          0521  2 IF NOT .status THEN
;  526          0522  2     RETURN .status;
;  527          0523  2 !
;  528          0524  2 ! The load is successfully finished.  Build the Parameter Load with Transfer
;  529          0525  2 ! address (PLT) message.  This message tells the target what address to start
```

```
;   530          0526  2 ! executing the image just loaded.
;   531          0527  2 !
;   532          0528  2 mom$bldmopplt (snddsc, .loadnum, .mom$l_transfer);
;   533          0529  2 !
;   534          0530  2 ! The newer NI loaders return a Request Memory Load message (with the load
;   535          0531  2 ! number = the last load frame + 2) as an acknowledgment to the Parameter
;   536          0532  2 ! Load with Transfer (PLT) message.  In the case of the tertiary, set up to
;   537          0533  2 ! skip over this message and keep looking for the request for the operating
;   538          0534  2 ! system.  In the case of the operating system, receipt of the RML indicates
;   539          0535  2 ! that the load is complete.
;   540          0536  2 !
;   541          0537  2 IF .mom$ab_service_data [svd$gk_pcno_sty, svd$l_param] EQL
;   542          0538  2                                                 nma$c_soft_terl THEN
;   543          0539  2     BEGIN
;   544          0540  2     PLT_response <0,8> = mop$_fct_rml;
;   545          0541  2     PLT_response + 1 <0,8> = .loadnum + 1;
;   546          0542  2     skip_msg_dsc_addr = PLT_response_dsc;
;   547          0543  2     END
;   548          0544  2 ELSE
;   549          0545  2     skip_msg_dsc_addr = 0;
;   550          0546  2 DECR retry FROM 4 TO 0 DO
;   551          0547  2     BEGIN
;   552          0548  2     status = mom$mopsndrcv (mom$ab_cib, snddsc,
;   553          0549  2                             mom$ab_cib, mom$gq_mop_rcv_buf_dsc,
;   554          0550  2                             msgdsc [0],
;   555          0551  2                             .skip_msg_dsc_addr);
;   556          0552  2     mom$chk_mop_error (.status);
;   557          0553  2     !
;   558          0554  2     ! A response was successfully received.  If it's another request for the
;   559          0555  2     ! PLT message (it's really a request for the last load frame + 1),
;   560          0556  2     ! retransmit the PLT.
;   561          0557  2     !
;   562          0558  2     IF (.mom$gq_mop_rcv_buf_dsc [0] LSS 2) OR
;   563          0559  2        (.mom$ab_mop_rcv_buf <0,8> NEQ mop$_fct_rml) OR
;   564          0560  2        (.mom$ab_mop_rcv_buf+1 <0,8> NEQ .loadnum) THEN
;   565          0561  2         EXITLOOP;
;   566          0562  2     status = failure;
;   567          0563  2     END;
;   568          0564  2 RETURN .status
;   569          0565  2
;   570          0566  1 END;                                ! End of mom_mblkload


                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                                00000002  0010C P.AAR:  .LONG   2
                                00000000' 00110         .ADDRESS PLT_RESPONSE

                                                .PSECT  $OWN$,NOEXE,2

                                        00C2C PLT_RESPONSE:
                                                .BLKB   4

                                     PLT_RESPONSE_DSC=   P.AAR
```

```
                                                        .PSECT  $CODE$,NOWRT,2

                                                03FC 00000 MOM_MBLKLOAD:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                            ; 0470
                                    59 00000000G   EF   9E 00002        MOVAB   MOM$AB_CIB, R9
                                    58 00000000G   EF   9E 00009        MOVAB   MOM$GQ_MOP_RCV_BUF_DSC, R8
                                    57 00000000G   EF   9E 00010        MOVAB   MOM$AB_MOP_RCV_BUF, R7
                                    56 00000000'   EF   9E 00017        MOVAB   MOM$W_FIRST_LOAD_FRAME, R6
                                    5E             0C   C2 0001E        SUBL2   #12, SP
                                    52       08    AC   D0 00021        MOVL    MSGDSC, R2                              ; 0509
                                       04   A2          67   9E 00025        MOVAB   MOM$AB_MOP_RCV_BUF, 4(R2)
                                    66             01   D0 00029        MOVL    #1, MOM$W_FIRST_LOAD_FRAME              ; 0516
                                10 00000000G   EF   06   E1 0002C        BBC     #6, MOM$GL_SERVICE_FLAGS, 1$            ; 0517
                                          4004     8F   BB 00034        PUSHR   #^M<R2,SP>                              ; 0518
                                            04     AC   DD 00038        PUSHL   LOADFLAG
                                00000000V   EF             03   FB 0003B        CALLS   #3, MOM_LOAD_CC_FILE
                                                   0E   11 00042        BRB     2$
                                          4004     8F   BB 00044 1$:     PUSHR   #^M<R2,SP>                              ; 0520
                                            04     AC   DD 00048        PUSHL   LOADFLAG
                                00000000V   EF             03   FB 0004B        CALLS   #3, MOM_LOAD_SYS_FILE
                                    54             50   D0 00052 2$:     MOVL    R0, STATUS
                                    68             54   E9 00055        BLBC    STATUS, 6$                              ; 0521
                                              FB   A6   DD 0005B        PUSHL   MGM$L_TRANSFER                          ; 0528
                                              04   AE   DD 0005B        PUSHL   LOADNUM
                                              0C   AE   9F 0005E        PUSHAB  SNDDSC
                                00000000G   EF             03   FB 00061        CALLS   #3, MOM$BLDMOPPLT
                                    01 00000000*   EF   D1 00068        CMPL    <<MOM$AB_SERVICE_DATA+<SVD$GK_PCNO_STY*137>-    ; 0537
                                                                                >+9>, #1
                                                   14   12 0006F        BNEQ    3$
                                          0C18     C6   0A   90 00071        MOVB    #10, PLT_RESPONSE                       ; 0540
                                    0C19   C6             6E   01   C1 00076        ADDL3   #1, LOADNUM, PLT_RESPONSE+1             ; 0541
                                    55 00000000'   EF   9E 0007C        MOVAB   PLT_RESPONSE_DSC, SKIP_MSG_DSC_ADDR     ; 0542
                                                   02   11 00083        BRB     4$                                      ; 0537
                                                   55   D4 00085 3$:     CLRL    SKIP_MSG_DSC_ADDR                       ; 0545
                                    53             04   D0 00087 4$:     MOVL    #4, RETRY                               ; 0551
                                                   24   BB 0008A 5$:     PUSHR   #^M<R2,R5>                              ; 0550
                                                   58   DD 0008C        PUSHL   R8                                      ; 0548
                                                   59   DD 0008E        PUSHL   R9
                                                     14   AE   9F 00090        PUSHAB  SNDDSC
                                                   59   DD 00093        PUSHL   R9
                                      00000000G    EF             06   FB 00095        CALLS   #6, MOM$MOPSNDRCV               ; 0550
                                    54             50   D0 0009C        MOVL    R0, STATUS
                                                   54   DD 0009F        PUSHL   STATUS                                  ; 0552
                                      00000000G    EF             01   FB 000A1        CALLS   #1, MOM$CHK_MOP_ERROR
                                    02             68   D1 000A8        CMPL    MOM$GQ_MOP_RCV_BUF_DSC, #2              ; 0558
                                                   13   19 000AB        BLSS    6$
                                    0A             67   91 000AD        CMPB    MOM$AB_MOP_RCV_BUF, #10                 ; 0559
                                                   0E   12 000B0        BNEQ    6$
                                 50                67   01   C1 000B2        ADDL3   #1, MOM$AB_MOP_RCV_BUF, R0              ; 0560
                                    6E             50   D1 000B6        CMPL    R0, LOADNUM
                                                   05   12 000B9        BNEQ    6$
                                                   54   D4 000BB        CLRL    STATUS                                  ; 0562
                                    CA             53   F4 000BD        SOBGEQ  RETRY, 5$                               ; 0546
                                    50             54   D0 000C0 6$:     MOVL    STATUS, R0                              ; 0564
                                                        04   000C3        RET                                             ; 0566

; Routine Size:  196 bytes,    Routine Base:  $CODE$ + 02C1
```

```
;  571          0567  1
```

```
;  573     0568  1 %SBTTL 'mom_load_sys_file      Perform load of system code'
;  574     0569  1 ROUTINE mom_load_sys_file (loadflag, recv_msg_dsc, final_loadnum) =
;  575     0570  1
;  576     0571  1 !++
;  577     0572  1 ! FUNCTIONAL DESCRIPTION:
;  578     0573  1 !
;  579     0574  1 !       This routine performs a multiblock system load of system
;  580     0575  1 !       code.  The file format system images is different than that
;  581     0576  1 !       for console carrier (see next routine).  The image is loaded
;  582     0577  1 !       into the target's memory contiguously, so there is only an
;  583     0578  1 !       address (supplied in the file header) specifying which address
;  584     0579  1 !       to begin loading the image.
;  585     0580  1 !
;  586     0581  1 ! FORMAL PARAMETERS:
;  587     0582  1 !       LOADFLAG        Address of load retry flag (TRUE=>if load failed
;  588     0583  1 !                       it failed on the first message exchange).
;  589     0584  1 !       RECV_MSG_DSC    Address of descriptor for received MOP message.
;  590     0585  1 !       FINAL_LOADNUM   Load number of last load frame + 1.  Returned to
;  591     0586  1 !                       caller to be used in the Parameter Load with
;  592     0587  1 !                       Transfer Address MOP message.
;  593     0588  1 !
;  594     0589  1 ! ROUTINE VALUE:
;  595     0590  1 ! COMPLETION CODES:
;  596     0591  1 !
;  597     0592  1 !       Signal errors.
;  598     0593  1 !
;  599     0594  1 !--
;  600     0595  1
;  601     0596  2 BEGIN
;  602     0597  2
;  603     0598  2 MAP
;  604     0599  2     recv_msg_dsc: REF BBLOCK;
;  605     0600  2
;  606     0601  2 LOCAL
;  607     0602  2     bufptr,
;  608     0603  2     len,
;  609     0604  2     loadnum : BYTE,
;  610     0605  2     loadblkcnt,
;  611     0606  2     loadbytcnt,
;  612     0607  2     blocks_left,                ! 64 byte blocks of data left in read buffer.
;  613     0608  2     ptr,
;  614     0609  2     snddsc  : VECTOR [2],
;  615     0610  2     status;
;  616     0611  2
;  617     0612  2 !
;  618     0613  2 ! LOADNUM is defined as a byte to correspond to the size of the field in the
;  619     0614  2 ! MOP message.  This field will overflow when it gets to load number 256 so
;  620     0615  2 ! it will go back to zero.  Overflow must be guaranteed in order for a load
;  621     0616  2 ! to succeed so great care should be taken to avoid BLISS optimizations that
;  622     0617  2 ! could change this situation.  Be especially careful if any compare or
;  623     0618  2 ! increment operations are modified.
;  624     0619  2 !
;  625     0620  2 loadnum = 0;
;  626     0621  2 !
;  627     0622  2 ! Load every block in the image.
;  628     0623  2 !
;  629     0624  2 WHILE .mom$l_loadsize GTR 0 DO
```

```
;   630          0625  3      BEGIN
;   631          0626  3
;   632          0627  3      status = success;                        ! Reset the status code
;   633          0628  3      !
;   634          0629  3      ! Read a block (record) from the file.
;   635          0630  3      !
;   636          0631  3      mom_readloadfile (mom$q_readbfdsc);
;   637          0632  3      !
;   638          0633  3      ! Load the image block (in one or more 64-byte pieces).
;   639          0634  3      !
;   640          0635  3      bufptr = mom$t_loadbuffer;
;   641          0636  3
;   642          0637  3      INCR i FROM 0 TO .mom$l_blkcnt - 1 BY mom$k_segblkcnt DO
;   643          0638  4          BEGIN
;   644          0639  4          !
;   645          0640  4          ! If the data left in the read buffer is less than the MOP transmit
;   646          0641  4          ! size (MOM$K_SEGBLKCNT * 64), send the data that's left.
;   647          0642  4          !
;   648          0643  4          blocks_left = .mom$l_blkcnt - .i;
;   649          0644  4          IF .blocks_left LSS mom$k_segblkcnt THEN
;   650          0645  4              loadblkcnt = .blocks_left
;   651          0646  4          ELSE
;   652          0647  4              loadblkcnt = mom$k_segblkcnt;
;   653          0648  4
;   654          0649  4          !
;   655          0650  4          ! Calculate the actual byte count of the data to be loaded.
;   656          0651  4          !
;   657          0652  4          loadbytcnt = .loadblkcnt * 64;
;   658          0653  4          !
;   659          0654  4          ! Build the MOP memory load message in the buffer around the
;   660          0655  4          ! image data.
;   661          0656  4          !
;   662          0657  4          ptr = .bufptr;
;   663          0658  4
;   664          0659  4          CH$WCHAR_A (mop$_fct_mld, ptr);          ! Function code
;   665          0660  4          CH$WCHAR_A (.loadnum, ptr);              ! Load number
;   666          0661  4          loadnum = .loadnum + 1;                  ! Increment load number
;   667          0662  4          ptr = CH$MOVE (4, mom$l_baseadr, .ptr); ! Base address
;   668          0663  4          ptr = .ptr + .loadbytcnt;                ! Skip image data
;   669          0664  4
;   670          0665  4          snddsc [1] = .bufptr;
;   671          0666  4          snddsc [0]  = .ptr - .bufptr;
;   672          0667  4          !
;   673          0668  4          ! Transmit the load data to the target node and receive a response.
;   674          0669  4          !
;   675          0670  4          status = mom_xmit_load_frame (.loadflag, snddsc, .recv_msg_dsc);
;   676          0671  4          IF NOT .status THEN
;   677          0672  4              EXITLOOP;
;   678          0673  4          !
;   679          0674  4          ! Decrement the number of blocks remaining to be loaded.
;   680          0675  4          !
;   681          0676  4          mom$l_loadsize = .mom$l_loadsize - .loadblkcnt;
;   682          0677  4          mom$l_baseadr = .mom$l_baseadr + .loadbytcnt;
;   683          0678  4          bufptr = .bufptr + .loadbytcnt;
;   684          0679  4
;   685          0680  3          END;
;   686          0681  3
```

```
;  687          0682  3      IF NOT .status THEN EXITLOOP;
;  688          0683  3
;  689          0684  2      END;
;  690          0685  2 .final_loadnum = .loadnum;
;  691          0686  2 RETURN .status;
;  692          0687  1 END;                ! of mom_load_sys_file



                                                OFFC 00000 MOM_LOAD_SYS_FILE:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11                    ; 0569
                                5B 00000000'    EF  9E 00002       MOVAB   MOM$L_LOADSIZE, R11
                                5E              08  C2 00009       SUBL2   #8, SP
                                                58  94 0000C       CLRB    LOADNUM                                         ; 0620
                                                6B  D5 0000E 1$:   TSTL    MOM$L_LOADSIZE                                  ; 0624
                                                78  15 00010       BLEQ    6$
                                55              01  D0 00012       MOVL    #1, STATUS                                      ; 0627
                                   00000000'    EF  9F 00015       PUSHAB  MOM$Q_READBFDSC                                 ; 0631
                     00000000V  EF              01  FB 0001B       CALLS   #1, MOM_READLOADFILE
                                53      0610    CB  9E 00022       MOVAB   MOM$T_LOADBUFFER, BUFPTR                        ; 0635
                   5A       FC  AB              01  C3 00027       SUBL3   #1, MOM$L_BLKCNT, R10                           ; 0637
                                54              04  CE 0002C       MNEGL   #4, I
                                                50  11 0002F       BRB     5$
                   59       FC  AB              54  C3 00031 2$:   SUBL3   I, MOM$L_BLKCNT, BLOCKS_LEFT                    ; 0643
                                04              59  D1 00036       CMPL    BLOCKS_LEFT, #4                                 ; 0644
                                                05  18 00039       BGEQ    3$
                                57              59  D0 0003B       MOVL    BLOCKS_LEFT, LOADBLKCNT                         ; 0645
                                                03  11 0003E       BRB     4$
                                57              04  D0 00040 3$:   MOVL    #4, LOADBLKCNT                                  ; 0647
                         56     57              06  78 00043 4$:   ASHL    #6, LOADBLKCNT, LOADBYTCNT                      ; 0652
                                52              53  D0 00047       MOVL    BUFPTR, PTR                                     ; 0657
                                82              02  90 0004A       MOVB    #2, (PTR)+                                      ; 0659
                                82              58  90 0004D       MOVB    LOADNUM, (PTR)+                                 ; 0660
                                                58  96 00050       INCB    LOADNUM                                         ; 0661
                                82      F8      AB  D0 00052       MOVL    MOM$L_BASEADR, (PTR)+                           ; 0662
                                52              56  C0 00056       ADDL2   LOADBYTCNT, PTR                                 ; 0663
                             04 AE              53  D0 00059       MOVL    BUFPTR, SNDDSC+4                                ; 0665
                         6E     52              53  C3 0005D       SUBL3   BUFPTR, PTR, SNDDSC                             ; 0666
                                            08  AC  DD 00061       PUSHL   RECV_MSG_DSC                                    ; 0670
                                            04  AE  9F 00064       PUSHAB  SNDDSC
                                            04  AC  DD 00067       PUSHL   LOADFLAG
                        00000000V           EF  03  FB 0006A       CALLS   #3, MOM_XMIT_LOAD_FRAME
                                55              50  D0 00071       MOVL    R0, STATUS
                                13              55  E9 00074       BLBC    STATUS, 6$                                      ; 0671
                                6B              57  C2 00077       SUBL2   LOADBLKCNT, MOM$L_LOADSIZE                      ; 0676
                             F8 AB              56  C0 0007A       ADDL2   LOADBYTCNT, MOM$L_BASEADR                       ; 0677
                                53              56  C0 0007E       ADDL2   LOADBYTCNT, BUFPTR                              ; 0678
                FFAA         54     04          5A  F1 00081 5$:   ACBL    R10, #4, I, 2$                                  ; 0637
                                84              55  E8 00087       BLBS    STATUS, 1$                                      ; 0682
                             0C BC              58  9A 0008A 6$:   MOVZBL  LOADNUM, @FINAL_LOADNUM                        ; 0685
                                50              55  D0 0008E       MOVL    STATUS, R0                                      ; 0686
                                                    04 00091       RET                                                     ; 0687

; Routine Size:  146 bytes,    Routine Base:  $CODE$ + 0385
```

```
;  694     0688  1 %SBTTL 'mom_load_cc_file      Perform load of console carrier code'
;  695     0689  1 ROUTINE mom_load_cc_file (loadflag, recv_msg_dsc, final_loadnum) =
;  696     0690  1
;  697     0691  1 !++
;  698     0692  1 ! FUNCTIONAL DESCRIPTION:
;  699     0693  1 !
;  700     0694  1 !       This routine performs a multiblock system load of the console
;  701     0695  1 !       carrier code.  The file format for console carrier is different
;  702     0696  1 !       from other load images because it is not necessarily into the
;  703     0697  1 !       target in contiguous memory locations.  Therefore the load
;  704     0698  1 !       "records" contain a target memory address and record length.
;  705     0699  1 !
;  706     0700  1 !      Absolute Loader Format for storing UNA Microcode on a Host
;  707     0701  1 !                       20-Aug-82     Al MacInnes
;  708     0702  1 !
;  709     0703  1 !       The following describes the planned file format for  the  Con-
;  710     0704  1 !       sole  Carrier  Server loadable microcode.  Please return any comments,
;  711     0705  1 !       problems, and/or suggestions.
;  712     0706  1 !
;  713     0707  1 !               The Absolute Loader format is planned to be used  for  storing
;  714     0708  1 !       UNA  loadable microcode, such as the Console Carrier Server, on a host
;  715     0709  1 !       system.  A file stored in this format would be read by a down-line lo-
;  716     0710  1 !       ader  task,  executing on some host, and loaded over the NI to the WCS
;  717     0711  1 !       and/or Link Memory of the destination UNA.  A file in Absolute  Loader
;  718     0712  1 !       format is comprised of some number of variable-length "records".  Each
;  719     0713  1 !       record specifies a 16-bit load address, followed  by  some  number  of
;  720     0714  1 !       bytes  which  are  to  be loaded into memory starting at that address.
;  721     0715  1 !       This should allow easy construction of  MOP  "Memory  Load"  messages.
;  722     0716  1 !       Also,  the  records are physically contiguous in the file, even if the
;  723     0717  1 !       "image" that is represented is not physically  contiguous.   In  other
;  724     0718  1 !       words,  there  are  no unused gaps in the file, even though, as in the
;  725     0719  1 !       case of a UNA loadable microcode image the code will reside in several
;  726     0720  1 !       discontiguous segments.
;  727     0721  1 !
;  728     0722  1 !               The Absolute Loader record format is as follows:
;  729     0723  1 !
;  730     0724  1 !       Byte    Contents
;  731     0725  1 !       ----    --------
;  732     0726  1 !
;  733     0727  1 !       0,1     always a binary word of "1"
;  734     0728  1 !       2,3     number of bytes in record (can be odd), from byte 0
;  735     0729  1 !               to last data byte, but excluding checksum byte
;  736     0730  1 !       4,5     PDP-11 load address
;  737     0731  1 !       6-last byte     image data
;  738     0732  1 !       last byte+1     checksum (XOR ?), can be ignored for our
;  739     0733  1 !                       purposes
;  740     0734  1 !
;  741     0735  1 !       Note:  a record with a byte count of "6" indicates the last record  of
;  742     0736  1 !       the  file.   If the load address of this last record is even, this re-
;  743     0737  1 !       presents a program transfer address.
;  744     0738  1 !
;  745     0739  1 ! FORMAL PARAMETERS:
;  746     0740  1 !       LOADFLAG        Address of load retry flag (TRUE=>if load failed
;  747     0741  1 !                       it failed on the first message exchange).
;  748     0742  1 !       RECV_MSG_DSC    Address of descriptor for received MOP message.
;  749     0743  1 !       FINAL_LOADNUM   Load number of last load frame + 1.  Returned to
;  750     0744  1 !                       caller to be used in the Parameter Load with
```

```
;  751     0745  1 !                          Transfer Address MOP message.
;  752     0746  1 !
;  753     0747  1 ! ROUTINE VALUE:
;  754     0748  1 ! COMPLETION CODES:
;  755     0749  1 !
;  756     0750  1 !       Signal errors.
;  757     0751  1 !
;  758     0752  1 !--
;  759     0753  1
;  760     0754  2 BEGIN
;  761     0755  2
;  762     0756  2 MAP
;  763     0757  2     recv_msg_dsc: REF BBLOCK;
;  764     0758  2
;  765     0759  2 MACRO
;  766     0760  2     !
;  767     0761  2     ! Console carrier load file definitons.  There are variable length "records"
;  768     0762  2     ! in the file that can be loaded to noncontiguous memory areas in the target.
;  769     0763  2     !
;  770     0764  2     cc_head =    0,0,16,0%,     ! load frame header word (always = 0001)
;  771     0765  2     cc_rec_len =  2,0,16,0%,    ! load frame data length
;  772     0766  2     cc_load_add = 4,0,16,0%,    ! load frame target address
;  773     0767  2
;  774     0768  2     mld_code =   0,0,8,0%,      ! MOP memory load message function code
;  775     0769  2     mld_load_num =1,0,8,0%,     ! MOP memory load message load number
;  776     0770  2     mld_add     =2,0,32,0%;     ! MOP memory load message target address
;  777     0771  2
;  778     0772  2 LOCAL
;  779     0773  2     buf_ptr: REF BBLOCK,
;  780     0774  2     load_rec_len,
;  781     0775  2     load_num: BYTE,     ! Number of this load frame.  Used for checking load
;  782     0776  2                         ! frame sequence between MOM and the target.
;  783     0777  2     mld_msg_dsc: VECTOR [2],
;  784     0778  2     record_end,
;  785     0779  2     partial_record_len, ! length of a partial load frame at the end of a
;  786     0780  2                         ! record from the load file.
;  787     0781  2     msgsize,
;  788     0782  2     status;
;  789     0783  2
;  790     0784  2 !
;  791     0785  2 ! Read the first record from the console carrier load file.
;  792     0786  2 !
;  793     0787  2 mom_readloadfile (mom$q_readbfdsc);
;  794     0788  2 load_num = 0;
;  795     0789  2 buf_ptr = mom$t_readbuffer;
;  796     0790  2 !
;  797     0791  2 ! Load the console carrier image file to the target.
;  798     0792  2 !
;  799     0793  2 WHILE true DO
;  800     0794  3     BEGIN
;  801     0795  3     !
;  802     0796  3     ! Each record in the console carrier must start with a word of 1.
;  803     0797  3     ! Validate this to make sure the right file is being loaded.
;  804     0798  3     !
;  805     0799  3     IF .buf_ptr [cc_head] NEQ 1 THEN
;  806     0800  4         BEGIN
;  807     0801  4         mom$ab_msgblock [msb$l_flags]  = msb$m_msg_fld;
```

```
;   808     0802  4          mom$ab_msgblock [msb$b_code]   = nma$c_sts_fco;
;   809     0803  4          mom$ab_msgblock [msb$w_detail] = .mom$w_pgmdetail;
;   810     0804  4          mom$ab_msgblock [msb$l_text]   = mom$_invccfil;
;   811     0805  4          mom$bld_reply (mom$ab_msgblock, msgsize);
;   812     0806  4          $signal_msg (mom$ab_nice_xmit_buf, .msgsize);
;   813     0807  3          END;
;   814     0808  3      !
;   815     0809  3      ! The last record of the console carrier load file has a byte count of 6.
;   816     0810  3      ! Load console carrier records until it is found.
;   817     0811  3      !
;   818     0812  3      IF .buf_ptr [cc_rec_len] EQL 6 THEN
;   819     0813  3          EXITLOOP;
;   820     0814  3      !
;   821     0815  3      ! Save the load record length so it can be overwritten with the MOP
;   822     0816  3      ! Memory Load message header information and the MOP message transmitted
;   823     0817  3      ! directly from the read buffer.
;   824     0818  3      !
;   825     0819  3      load_rec_len = .buf_ptr [cc_rec_len];
;   826     0820  3      record_end = .buf_ptr + .load_rec_len + 1;
;   827     0821  3      If .record_end GTR mom$t_readbuffer + mom$k_loadbufsiz THEN
;   828     0822  3          !
;   829     0823  3          ! The load frame is partly in this record, and partly in the next one.
;   830     0824  3          ! Move the beginning of this load frame so, when the next file read
;   831     0825  3          ! is complete, the record is contiguous.
;   832     0826  3          !
;   833     0827  4          BEGIN
;   834     0828  4          partial_record_len = mom$t_readbuffer + mom$k_loadbufsiz - .buf_ptr;
;   835     0829  4          CH$MOVE (.partial_record_len,
;   836     0830  4                   .buf_ptr,
;   837     0831  4                   mom$t_readbuffer - .partial_record_len);
;   838     0832  4          !
;   839     0833  4          ! Get the next buffer from the load file.
;   840     0834  4          !
;   841     0835  4          mom_readloadfile (mom$g_readbfdsc);
;   842     0836  4          buf_ptr = mom$t_readbuffer - .partial_record_len;
;   843     0837  3          END;
;   844     0838  3      !
;   845     0839  3      ! Build the MOP message in the read buffer and transmit it to the target
;   846     0840  3      ! from the read buffer.  Overwrite the record byte count with the MOP
;   847     0841  3      ! Memory Load function code and load number.
;   848     0842  3      !
;   849     0843  3      buf_ptr [mld_code] = mop$_fct_mld;
;   850     0844  3      buf_ptr [mld_load_num] = .load_num;
;   851     0845  3      buf_ptr [mld_add] = .buf_ptr [cc_load_add];
;   852     0846  3      load_num = .load_num + 1;
;   853     0847  3      mld_msg_dsc [0] = .load_rec_len;
;   854     0848  3      mld_msg_dsc [1] = .buf_ptr;
;   855     0849  3      !
;   856     0850  3      ! If the console carrier load record won't fit in the load buffer
;   857     0851  3      ! (this size is fixed when the load is initiated), signal an "image
;   858     0852  3      ! record size" error
;   859     0853  3      !
;   860     0854  3      IF .mld_msg_dsc [0] GTR (mom$k_segblkcnt * 64) THEN
;   861     0855  4          BEGIN
;   862     0856  4          mom$ab_msgblock [msb$l_flags]  = msb$m_det_fld OR
;   863     0857  4                                           msb$m_msg_fld;
;   864     0858  4          mom$ab_msgblock [msb$b_code]   = nma$c_sts_fio;
```

```
;   865         0859  4          mom$ab_msgblock [msb$w_detail] = .mom$ab_service_data [svd$gk_pcno_sty,
;   866         0860  4                                                                svd$l_param];
;   867         0861  4          mom$ab_msgblock [msb$l_text] = mom$_imgrecsiz;
;   868         0862  4          mom$bld_reply (mom$ab_msgblock, msgsize);
;   869         0863  4          $signal_msg (mom$ab_nice_xmit_buf, .msgsize);
;   870         0864             END;
;   871         0865         !
;   872         0866         ! Send the MOP Memory Load message to the target and get a response.
;   873         0867         !
;   874         0868         status = mom_xmit_load_frame (.loadflag, mld_msg_dsc, .recv_msg_dsc);
;   875         0869         IF NOT .status THEN
;   876         0870             EXITLOOP;
;   877         0871         !
;   878         0872         ! Point to next "record" in the buffer, skipping the checksum byte at
;   879         0873         ! the end which is not included in the record length field.
;   880         0874         !
;   881         0875         buf_ptr = .buf_ptr + .load_rec_len + 1;
;   882         0876         END;
;   883         0877     !
;   884         0878     ! If the load address of the last record is even, use it as the transfer
;   885         0879     ! address.
;   886         0880     !
;   887         0881     IF NOT .buf_ptr [cc_load_add] THEN
;   888         0882         mom$l_transfer = .buf_ptr [cc_load_add]
;   889         0883     ELSE
;   890         0884         mom$l_transfer = 0;
;   891         0885     .final_loadnum = .load_num;
;   892         0886     RETURN .status;
;   893         0887     END;                ! of mom_load_cc_file


                                                OFFC 00000 MOM_LOAD_CC_FILE:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11                ; 0689
                                    5E                  10  C2 00002           SUBL2   #16, SP                         ;
                                            00000000'   EF  9F 00005           PUSHAB  MOM$Q_READBFDSC                 ; 0787
                                00000000V   EF          01  FB 0000B           CALLS   #1, MOM_READLOADFILE            ;
                                                        6E  94 00012           CLRB    LOAD_NUM                        ; 0788
                                            56 00000000'EF  9E 00014           MOVAB   MOM$T_READBUFFER, BUF_PTR       ; 0789
                                            01          66  B1 0001B 1$:       CMPW    (BUF_PTR), #1                   ; 0799
                                                        4A  13 0001E           BEQL    2$                              ;
                                00000000G   EF          04  D0 00020           MOVL    #4, MOM$AB_MSGBLOCK             ; 0801
                                00000000G   EF          0E  8E 00027           MNEGB   #14, MOM$AB_MSGBLOCK+4          ; 0802
                                00000000G   EF  00000000'EF  B0 0002E           MOVW    MOM$W_PGMDETAIL, MOM$AB_MSGBLOCK+8 ; 0803
                                00000000G   EF  00000000G8F  D0 00039           MOVL    #MOM$_INVCCFIL, MOM$AB_MSGBLOCK+12 ; 0804
                                                      04    AE  9F 00044           PUSHAB  MSGSIZE                         ; 0805
                                                00000000G   EF  9F 00047           PUSHAB  MOM$AB_MSGBLOCK                 ;
                                00000000G   EF          02  FB 0004D           CALLS   #2, MOM$BLD_REPLY               ;
                                                      04    AE  DD 00054           PUSHL   MSGSIZE                         ; 0806
                                                00000000G   EF  9F 00057           PUSHAB  MOM$AB_NICE_XMIT_BUF            ;
                                                02070000    8F  DD 0005D           PUSHL   #34013T84                       ;
                                00000000G   00          03  FB 00063           CALLS   #3, LIB$SIGNAL                  ;
                                            06        02    A6  B1 0006A 2$:       CMPW    2(BUF_PTR), #6                  ; 0812
                                                        03  12  0006E           BNEQ    3$                              ;
                                                      00C6  31  00070           BRW     6$                              ;
```

```
                        58        02   A6  3C 00073 3$:      MOVZWL  2(BUF_PTR), LOAD_REC_LEN                             ; 0819
                        58        01 A846  9E 00077          MOVAB   1(LOAD_REC_LEN)[BUF_PTR], RECORD_END                 ; 0820
                        50  00000000'  EF  9E 0007C          MOVAB   MOM$T_READBUFFER+1536, R0                            ; 0821
                        50             5B  D1 00083          CMPL    RECORD_END, R0                                       ;
                                       29  15 00086          BLEQ    4$                                                   ;
                        50  00000000'  EF  9E 00088          MOVAB   MOM$T_READBUFFER+1536, R0                            ; 0828
             59         50             56  C3 0008F          SUBL3   BUF_PTR, R0, PARTIAL_RECORD_LEN                      ;
                        57  00000000'  EF  9E 00093          MOVAB   MOM$T_READBUFFER, R7                                 ; 0829
                        57             59  C2 0009A          SUBL2   PARTIAL_RECORD_LEN, R7                               ; 0831
             67         66             59  28 0009D          MOVC3   PARTIAL_RECORD_LEN, (BUF_PTR), (R7)                  ;
                            00000000'  EF  9F 000A1          PUSHAB  MOM$Q_READBFDSC                                      ; 0835
             00000000V  EF             01  FB 000A7          CALLS   #1, MOM_READLOADFILE                                 ;
                        56             57  D0 000AE          MOVL    R7, BUF_PTR                                          ; 0836
                        66             02  90 000B1 4$:      MOVB    #2, (BUF_PTR)                                        ; 0843
                  01    A6             6E  90 000B4          MOVB    LOAD_NUM, 1(BUF_PTR)                                 ; 0844
                  02    A6        04   A6  3C 000B8          MOVZWL  4(BUF_PTR), 2(BUF_PTR)                               ; 0845
                                       6E  96 000BD          INCB    LOAD_NUM                                             ; 0846
                  08    AE             58  D0 000BF          MOVL    LOAD_REC_LEN, MLD_MSG_DSC                            ; 0847
                  0C    AE             56  D0 000C3          MOVL    BUF_PTR, MLD_MSG_DSC+4                               ; 0848
             00000100   8F        08   AE  D1 000C7          CMPL    MLD_MSG_DSC, #256                                    ; 0854
                                       4A  15 000CF          BLEQ    5$                                                   ;
             00000000G  EF             06  D0 000D1          MOVL    #6, MOM$AB_MSGBLOCK                                  ; 0856
             00000000G  EF             12  8E 000D8          MNEGB   #18, MOM$AB_MSGBLOCK+4                               ; 0858
             00000000G  EF  00000000*  EF  B0 000DF          MOVW    <<MOM$AB_SERVICE_DATA+<SVD$GK_PCNO_STY*137>-         ; 0859
                                                                     >+9>, MOM$AB_MSGBLOCK+8                              ;
             00000000G  EF  00000000G  8F  D0 000EA          MOVL    #MOM$_IMGRECSIZ, MOM$AB_MSGBLOCK+12                  ; 0861
                                  04   AE  9F 000F5          PUSHAB  MSGSIZE                                              ; 0862
                            00000000G  EF  9F 000F8          PUSHAB  MOM$AB_MSGBLOCK                                      ;
             00000000G  EF             02  FB 000FE          CALLS   #2, MOM$BLD_REPLY                                    ;
                                  04   AE  DD 00105          PUSHL   MSGSIZE                                              ; 0863
                            00000000G  EF  9F 00108          PUSHAB  MOM$AB_NICE_XMIT_BUF                                 ;
                            02070000   8F  DD 0010E          PUSHL   #34013184                                            ;
             00000000G  00             03  FB 00114          CALLS   #3, LIB$SIGNAL                                       ;
                                  08   AC  DD 0011B 5$:      PUSHL   RECV_MSG_DSC                                         ; 0868
                                  0C   AE  9F 0011E          PUSHAB  MLD_MSG_DSC                                          ;
                                  04   AC  DD 00121          PUSHL   LOADFLAG                                             ;
             00000000V  EF             03  FB 00124          CALLS   #3, MOM_XMIT_LOAD_FRAME                              ;
                        5A             50  D0 0012B          MOVL    R0, STATUS                                           ;
                        08             5A  E9 0012E          BLBC    STATUS, 6$                                           ; 0869
                        56        01 A846  9E 00131          MOVAB   1(LOAD_REC_LEN)[BUF_PTR], BUF_PTR                    ; 0875
                                     FEE2  31 00136          BRW     1$                                                   ; 0793
                        0A        04   A6  E8 00139 6$:      BLBS    4(BUF_PTR), 7$                                       ; 0881
             00000000'  EF        04   A6  3C 0013D          MOVZWL  4(BUF_PTR), MOM$L_TRANSFER                           ; 0882
                                       06  11 00145          BRB     8$                                                   ;
                            00000000'  EF  D4 00147 7$:      CLRL    MOM$L_TRANSFER                                       ; 0884
                  0C    BC             6E  9A 0014D 8$:      MOVZBL  LOAD_NUM, @FINAL_LOADNUM                             ; 0885
                        50             5A  D0 00151          MOVL    STATUS, R0                                           ; 0886
                                           04 00154          RET                                                          ; 0887

; Routine Size:  341 bytes,    Routine Base:  $CODE$ + 0417
```

```
;   895          0888  1 %SBTTL 'mom_xmit_load_frame     Transmit multiblock load frame to target'
;   896          0889  1 ROUTINE mom_xmit_load_frame  (loadflag, xmit_msg_dsc, recv_msg_dsc) =
;   897          0890  1
;   898          0891  1 !++
;   899          0892  1 ! FUNCTIONAL DESCRIPTION:
;   900          0893  1 !
;   901          0894  1 !       This routine sends a single load frame to the target during
;   902          0895  1 !       a multiblock load sequence.
;   903          0896  1 !
;   904          0897  1 ! FORMAL PARAMETERS:
;   905          0898  1 !
;   906          0899  1 !       LOADFLAG        Address of load retry flag (TRUE=>if load failed
;   907          0900  1 !                       it failed on the first message exchange).
;   908          0901  1 !       XMIT_MSG_DSC    Address of descriptor of MOP messageto transmit.
;   909          0902  1 !       RCV_MSG_DSC     Address of descriptor for received MOP message.
;   910          0903  1 !
;   911          0904  1 ! ROUTINE VALUE:
;   912          0905  1 ! COMPLETION CODES:
;   913          0906  1 !
;   914          0907  1 !       Signal errors.
;   915          0908  1 !
;   916          0909  1 !--
;   917          0910  1
;   918          0911  2 BEGIN
;   919          0912  2
;   920          0913  2 MAP
;   921          0914  2     xmit_msg_dsc: REF VECTOR,
;   922          0915  2     recv_msg_dsc: REF VECTOR;
;   923          0916  2
;   924          0917  2 LOCAL
;   925          0918  2     skip_msg_dsc_addr,
;   926          0919  2     next_loadnum : BYTE,
;   927          0920  2     status;
;   928          0921  2
;   929          0922  2 DECR retry FROM 4 TO 0 DO
;   930          0923  3     BEGIN
;   931          0924  3     !
;   932          0925  3     ! For NI circuits, program load requests are retransmitted if no
;   933          0926  3     ! response is received within a specified time.  If this the first
;   934          0927  3     ! load frame, set up to skip them, in case there are a number of
;   935          0928  3     ! these messages backed up on the circuit.
;   936          0929  3     !
;   937          0930  3     IF .mom$w_first_load_frame THEN
;   938          0931  3         skip_msg_dsc_addr = mom$gq_mop_msg_dsc
;   939          0932  3     ELSE
;   940          0933  3         skip_msg_dsc_addr = 0;
;   941          0934  3     status = mom$mopsndrcv (mom$ab_cib, .xmit_msg_dsc,
;   942          0935  3                             mom$ab_cib, mom$gq_mop_rcv_buf_dsc,
;   943          0936  3                             recv_msg_dsc [0],
;   944          0937  3                             .skip_msg_dsc_addr);
;   945          0938  3     IF NOT .status THEN
;   946          0939  4         BEGIN
;   947          0940  4         IF ..loadflag THEN
;   948          0941  5             BEGIN
;   949          0942  5             mom$ab_msgblock [msb$l_flags] = 0;
;   950          0943  5             mom$ab_msgblock [msb$b_code]  = nma$c_sts_lco;
;   951          0944  5             EXITLOOP;
```

```
;  952          0945  5                END
;  953          0946  4            ELSE
;  954          0947  4                mom$chk_mop_error (.status);
;  955          0948  4            END;
;  956          0949  3        !
;  957          0950  3        ! Verify the response message from the target node.  It must
;  958          0951  3        ! be a MOP request memory load message.
;  959          0952  3        !
;  960          0953  3        IF (.recv_msg_dsc [0] lss 2)
;  961          0954  3            OR (CASRCHAR (mom$ab_mop_rcv_buf) NEQ mop$_fct_rml) THEN
;  962          0955  4            BEGIN
;  963          0956  4            mom$ab_msgblock [msb$l_flags] = 0;
;  964          0957  4            mom$ab_msgblock [msb$b_code]  = nma$c_sts_ipr;
;  965          0958  4            status = failure;
;  966          0959  4            EXITLOOP;
;  967          0960  4            END;
;  968          0961  3        !
;  969          0962  3        ! If response message from the target node is requesting the
;  970          0963  3        ! the next load buffer, then don't retry.
;  971          0964  3        !
;  972          0965  3        next_loadnum = .(.xmit_msg_dsc [1] + 1)<0,8> +1;
;  973          0966  3        IF .(mom$ab_mop_rcv_buf + 1)<0,8> EQL .next_loadnum THEN
;  974          0967  4            BEGIN
;  975          0968  4            .loadflag = false;
;  976          0969  4            EXITLOOP;
;  977          0970  4            END;
;  978          0971  2        END;
;  979          0972  2    mom$w_first_load_frame = 0;
;  980          0973  2    RETURN .status;
;  981          0974  1    END;                        ! End of mom_xmit_load_frame


                                    03FC 00000 MOM_XMIT_LOAD_FRAME:
                                                       .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                    ; 0889
                          59 00000000'  EF  9E 00002           MOVAB   MOM$W_FIRST_LOAD_FRAME, R9
                          58 00000000G  EF  9E 00009           MOVAB   MOM$AB_CIB, R8
                          57 00000000G  EF  9E 00010           MOVAB   MOM$AB_MSGBLOCK, R7
                          52       08   AC  D0 00017           MOVL    XMIT_MSG_DSC, R2                        ; 0965
                          54           04  D0 0001B           MOVL    #4, RETRY
                          09           69  E9 0001E 1$:       BLBC    MOM$W_FIRST_LOAD_FRAME, 2$              ; 0930
                          55 00000000G  EF  9E 00021           MOVAB   MOM$GQ_MOP_MSG_DSC, SKIP_MSG_DSC_ADDR   ; 0931
                                       02  11 00028           BRB     3$
                                       55  D4 0002A 2$:       CLRL    SKIP_MSG_DSC_ADDR                       ; 0933
                                       55  DD 0002C 3$:       PUSHL   SKIP_MSG_DSC_ADDR                       ; 0937
                                    0C     AC  DD 0002E           PUSHL   RECV_MSG_DSC                        ; 0936
                                00000000G  EF  9F 00031           PUSHAB  MOM$GQ_MOP_RCV_BUF_DSC              ; 0934
                                       58  DD 00037           PUSHL   R8
                                       08  AC  DD 00039           PUSHL   XMIT_MSG_DSC                        ; 0936
                                       58  DD 0003C           PUSHL   R8                                  ; 0934
                      00000000G  EF                06  FB 0003E           CALLS   #6, MOM$MOPSNDRCV               ; 0936
                                 53                50  D0 00045           MOVL    R0, STATUS
                                 15                53  E8 00048           BLBS    STATUS, 5$                      ; 0938
                                 08          04    BC  E9 0004B           BLBC    @LOADFLAG, 4$                   ; 0940
                                                   67  D4 0004F           CLRL    MOM$AB_MSGBLOCK                 ; 0942
```

```
                          04   A7                 0A   8E 00051          MNEGB   #10, MOM$AB_MSGBLOCK+4                  ; 0943
                                                  3C   11 00055          BRB     9$                                      ; 0941
                                                  53   DD 00057 4$:      PUSHL   STATUS                                  ; 0947
                00000000G   EF                    01   FB 00059          CALLS   #1, MOM$CHK_MOP_ERROR
                            02        0C          BC   D1 00060 5$:      CMPL    @RECV_MSG_DSC, #2                       ; 0953
                                                  09   19 00064          BLSS    6$
                            0A 00000000G          EF   91 00066          CMPB    MOM$AB_MOP_RCV_BUF, #10                 ; 0954
                                                  0A   13 0006D          BEQL    7$
                                                  67   D4 0006F 6$:      CLRL    MOM$AB_MSGBLOCK                         ; 0956
                          04   A7                 11   8E 00071          MNEGB   #17, MOM$AB_MSGBLOCK+4                  ; 0957
                                                  53   D4 00075          CLRL    STATUS                                  ; 0958
                                                  1A   11 00077          BRB     9$                                      ; 0955
                            50        04          A2   D0 00079 7$:      MOVL    4(R2), R0                               ; 0965
                56          01        A0          01   81 0007D          ADDB3   #1, 1(R0), NEXT_LOADNUM
                            56 00000000G          EF   91 00082          CMPB    MOM$AB_MOP_RCV_BUF+1, NEXT_LOADNUM      ; 0966
                                                  05   12 00089          BNEQ    8$
                                      04          BC   D4 0008B          CLRL    @LOADFLAG                               ; 0968
                                                  03   11 0008E          BRB     9$                                      ; 0967
                            8B                    54   F4 00090 8$:      SOBGEQ  RETRY, 1$                               ; 0922
                                                  69   D4 00093 9$:      CLRL    MOM$W_FIRST_LOAD_FRAME                  ; 0972
                            50                    53   D0 00095          MOVL    STATUS, R0                              ; 0973
                                                       04 00098          RET                                             ; 0974

; Routine Size:  153 bytes,    Routine Base:  $CODE$ + 056C
```

```
;  983          0975  1 %SBTTL 'mom_secload       Perform secondary bootstrap load'
;  984          0976  1 ROUTINE mom_secload (loadflag, msgdsc) =
;  985          0977  1
;  986          0978  1 !++
;  987          0979  1 ! FUNCTIONAL DESCRIPTION:
;  988          0980  1 !
;  989          0981  1 !       This routine down line loads the secondary bootstrap loader to the
;  990          0982  1 !       target node.  It sends the entire load image in a single MOP
;  991          0983  1 !       message.  This is required by MOP to keep the primary boot as simple
;  992          0984  1 !       as possible.
;  993          0985  1 !
;  994          0986  1 ! FORMAL PARAMETERS:
;  995          0987  1 !
;  996          0988  1 !       LOADFLAG        Address of load retry flag (TRUE=>if load failed
;  997          0989  1 !                       it failed on the first message exchange).
;  998          0990  1 !       MSGDSC          Address of descriptor for received MOP message.
;  999          0991  1 !
; 1000          0992  1 ! IMPLICIT INPUTS:
; 1001          0993  1 !
; 1002          0994  1 !       NONE
; 1003          0995  1 !
; 1004          0996  1 ! IMPLICIT OUTPUTS:
; 1005          0997  1 !
; 1006          0998  1 !       NONE
; 1007          0999  1 !
; 1008          1000  1 ! ROUTINE VALUE:
; 1009          1001  1 ! COMPLETION CODES:
; 1010          1002  1 !
; 1011          1003  1 !       Signal errors.
; 1012          1004  1 !
; 1013          1005  1 ! SIDE EFFECTS:
; 1014          1006  1 !
; 1015          1007  1 !       NONE
; 1016          1008  1 !--
; 1017          1009  1
; 1018          1010  2 BEGIN
; 1019          1011  2
; 1020          1012  2 MAP
; 1021          1013  2     msgdsc : REF VECTOR;
; 1022          1014  2
; 1023          1015  2 LOCAL
; 1024          1016  2     load_byte_cnt,              ! Byte count of secondary boot loader image.
; 1025          1017  2     ptr,
; 1026          1018  2     snddsc : VECTOR [2],
; 1027          1019  2     status,
; 1028          1020  2     skip_msg_dsc_addr;
; 1029          1021  2 !
; 1030          1022  2 ! Check the load size.  The entire secondary loader image must fit in the
; 1031          1023  2 ! transmit buffer.  MOM$L_LOADSIZE is secondary bootstrap image size.  It
; 1032          1024  2 ! was obtained from the secondary boostrap file header, and is specified
; 1033          1025  2 ! in 32 word blocks.
; 1034          1026  2 !
; 1035          1027  2 load_byte_cnt = .mom$l_loadsize * 64;
; 1036          1028  2 !
; 1037          1029  2 ! If the byte count is slightly greater than 1500 because the loader took
; 1038          1030  2 ! the last 32 word block and went over the limit, truncate the length of
; 1039          1031  2 ! the loader down to fit into a single NI message.
```

```
;  1040          1032  2 !
;  1041          1033  2 IF .load_byte_cnt GTRU mom$k_maxsecsiz
;  1042          1034  2     AND .load_byte_cnt LEQU mom$k_loadbufsiz
;  1043          1035  2 THEN
;  1044          1036  2     load_byte_cnt = mom$k_maxsecsiz;
;  1045          1037  2 !
;  1046          1038  2 ! Make sure the message fits into a single NI message
;  1047          1039  2 !
;  1048          1040  2 IF .load_byte_cnt GTRU mom$k_loadbufsiz THEN
;  1049          1041  2     mom$error (nma$c_sts_fco, .mom$w_pgmdetail);
;  1050          1042  2 IF NOT .mom$gl_service_flags [mom$v_ni_circ] THEN
;  1051          1043  2     BEGIN
;  1052          1044  2     !
;  1053          1045  2     ! MOP specifies that the transfer address and image start address must be
;  1054          1046  2     ! 6.  For generality, add 6 to the values specified for these fields in
;  1055          1047  2     ! the secondary load file header.
;  1056          1048  2     !
;  1057          1049  2     mom$l_baseadr = .mom$l_baseadr + 6;
;  1058          1050  2     mom$l_transfer = .mom$l_transfer + 6;
;  1059          1051  2     END;
;  1060          1052  2 !
;  1061          1053  2 ! Read a block from the load image file.
;  1062          1054  2 !
;  1063          1055  2 mom_readloadfile (mom$q_readbfdsc);
;  1064          1056  2 !
;  1065          1057  2 ! Fill in the MOP message information.
;  1066          1058  2 !
;  1067          1059  2 ptr = mom$t_loadbuffer;
;  1068          1060  2
;  1069          1061  2 ch$wchar_a (mop$_fct_mlt, ptr); ! Function code
;  1070          1062  2 ch$wchar_a (0, ptr);           ! Load number
;  1071          1063  2 ptr = ch$move (4, mom$l_baseadr, .ptr); ! Load address (base)
;  1072          1064  2 ptr = .ptr + .load_byte_cnt;    ! Skip image data
;  1073          1065  2 ptr = ch$move (4, mom$l_transfer, .ptr); ! Transfer address
;  1074          1066  2
;  1075          1067  2 snddsc [0] = .ptr - mom$t_loadbuffer;
;  1076          1068  2 snddsc [1] = mom$t_loadbuffer;
;  1077          1069  2 msgdsc [1] = mom$ab_mop_rcv_buf;
;  1078          1070  2 !
;  1079          1071  2 ! Send the message and receive the response.  If the request for the secondary
;  1080          1072  2 ! was an NI multicast, MOM is essentially volunteering assistance.  Send the
;  1081          1073  2 ! secondary only once (as you would with an assistance volunteer), and if no
;  1082          1074  2 ! response is received, quit.  Some other host responded to the multicast
;  1083          1075  2 ! first.
;  1084          1076  2 !
;  1085          1077  2 IF .mom$gl_service_flags [mom$v_ni_volunteering] THEN
;  1086          1078  2     mom$ab_cib [cib$l_retry_cnt] = 1;
;  1087          1079  2 !
;  1088          1080  2 ! If it's an NI circuit, the target could have multicast the Program Load
;  1089          1081  2 ! Request more than once.  If so, skip over these messages until one is
;  1090          1082  2 ! received which is a response to the secondary loader.
;  1091          1083  2 !
;  1092          1084  2 IF .mom$gl_service_flags [mom$v_ni_circ] THEN
;  1093          1085  2     skip_msg_dsc_addr = mom$gq_mop_msg_dsc
;  1094          1086  2 ELSE
;  1095          1087  2     skip_msg_dsc_addr = 0;
;  1096          1088  2 status = mom$mopsndrcv (mom$ab_cib, snddsc,
```

```
;  1097          1089  2                          mom$ab_cib, mom$gq_mop_rcv_buf_dsc,
;  1098          1090  2                          msgdsc [0],
;  1099          1091  2                          .skip_msg_dsc_addr);
;  1100          1092  2
;  1101          1093  2                  ! If the receive failed and no messages had been previously exchanged
;  1102          1094  2                  ! then return the error status.  If the receive failed and some messages
;  1103          1095  2                  ! had been exchanged then signal a communications error to terminate
;  1104          1096  2                  ! the operation.
;  1105          1097  2                  !
;  1106          1098  2                  IF (NOT .status) AND (NOT ..loadflag) THEN
;  1107          1099  2                      mom$chk_mop_error (.status);
;  1108          1100  2                  !
;  1109          1101  2                  ! Restore retry count in case MOM was volunteering assistance and a response
;  1110          1102  2                  ! addressed directly to this node was received.  This means this node was
;  1111          1103  2                  ! chosen by the target to do the load.
;  1112          1104  2                  !
;  1113          1105  2                  mom$ab_cib [cib$l_retry_cnt] = 5;
;  1114          1106  2                  !
;  1115          1107  2                  ! If the target responded with a message addressed directly to this node,
;  1116          1108  2                  ! exit the volunteering state.  This node was chosen to perform the load.
;  1117          1109  2                  ! All further messages between MOM and the target will be non multicast.
;  1118          1110  2                  !
;  1119          1111  2                  IF .status THEN
;  1120          1112  3                      BEGIN
;  1121          1113  3                      IF NOT .mom$gl_service_flags [mom$v_ni_multicast] THEN
;  1122          1114  3                          mom$gl_service_flags [mom$v_ni_volunteering] = false
;  1123          1115  3                      ELSE
;  1124          1116  3                          !
;  1125          1117  3                          ! MOM got a multicast request from the target that wasn't a request
;  1126          1118  3                          ! for the secondary.  Quit.  Presumably the target will retransmit
;  1127          1119  3                          ! the request and MOM will get started up again in a context that it
;  1128          1120  3                          ! can process the request.
;  1129          1121  3                          !
;  1130          1122  3                          status = failure;
;  1131          1123  3                      END;
;  1132          1124  2                  RETURN .status
;  1133          1125  2
;  1134          1126  1  END;                                   ! End of mom_secload


                                                003C 00000 MOM_SECLOAD:
                                                                .WORD   Save R2,R3,R4,R5                                        ; 0976
                                    55 00000000G   EF  9E 00002         MOVAB   MOM$AB_CIB+18, R5                                   ;
                                    54 00000000G   EF  9E 00009         MOVAB   MOM$GL_SERVICE_FLAGS, R4                            ;
                                    53 00000000'   EF  9E 00010         MOVAB   MOM$T_LOADBUFFER, R3                                ;
                                                   5E  C2 00017         SUBL2   #8, SP                                              ;
                                52      F9F0   C3      06  78 0001A         ASHL    #6, MOM$L_LOADSIZE, LOAD_BYTE_CNT           ; 1027
                                   000005D0    8F      52  D1 00020         CMPL    LOAD_BYTE_CNT, #1488                        ; 1033
                                                       0E  1B 00027         BLEQU   1$                                          ;
                                   00000600    8F      52  D1 00029         CMPL    LOAD_BYTE_CNT, #1536                        ; 1034
                                                       05  1A 00030         BGTRU   1$                                          ;
                                               52      05D0    8F  3C 00032         MOVZWL  #1488, LOAD_BYTE_CNT                ; 1036
                                   00000600    8F      52  D1 00037 1$:      CMPL    LOAD_BYTE_CNT, #T536                        ; 1040
                                                       0F  1B 0003E         BLEQU   2$                                          ;
```

```
                          7E      F9F8    C3  3C 00040          MOVZWL  MOM$W_PGMDETAIL, -(SP)                          ; 1041
                          7E              0E  CE 00045          MNEGL   #14, =(SP)
                00000000G EF              02  FB 00048          CALLS   #2, MOM$ERROR
              0A          64              01  E0 0004F 2$:      BBS     #1, MOM$GL_SERVICE_FLAGS, 3$                  ; 1042
                    F9E8  C3              06  C0 00053          ADDL2   #6, MOM$L_BASEADR                             ; 1049
                    F9F4  C3              06  C0 00058          ADDL2   #6, MOM$L_TRANSFER                            ; 1050
                          00000000'       EF  9F 0005D 3$:      PUSHAB  MOM$Q_READBFDSC                               ; 1055
                00000000V EF              01  FB 00063          CALLS   #1, MOM_READLOADFILE
                          51              63  9E 0006A          MOVAB   MOM$T_LOADBUFFER, PTR                         ; 1059
                                          81  B4 0006D          CLRW    (PTR)+                                        ; 1061
                          81      F9E8    C3  D0 0006F          MOVL    MOM$L_BASEADR, (PTR)+                         ; 1063
                          51              52  C0 00074          ADDL2   LOAD_BYTE_CNT, PTR                            ; 1064
                          81      F9F4    C3  D0 00077          MOVL    MOM$L_TRANSFER, (PTR)+                        ; 1065
                          50              63  9E 0007C          MOVAB   MOM$T_LOADBUFFER, R0                          ; 1067
              6E          51              50  C3 0007F          SUBL3   R0, PTR, SNDDSC
                    04    AE              63  9E 00083          MOVAB   MOM$T_LOADBUFFER, SNDDSC+4                    ; 1068
                          50      08      AC  D0 00087          MOVL    MSGDSC, R0                                    ; 1069
                    04    A0 00000000G    EF  9E 0008B          MOVAB   MOM$AB_MOP_RCV_BUF, 4(R0)
                                          64  95 00093          TSTB    MOM$GL_SERVICE_FLAGS                          ; 1077
                                          03  18 00095          BGEQ    4$
                          65              01  D0 00097          MOVL    #1, MOM$AB_CIB+18                             ; 1078
              09          64              01  E1 0009A 4$:      BBC     #1, MOM$GL_SERVICE_FLAGS, 5$                  ; 1084
                          51 00000000G    EF  9E 0009E          MOVAB   MOM$GQ_MOP_MSG_DSC, SKIP_MSG_DSC_ADDR         ; 1085
                                          02  11 000A5          BRB     6$
                                          51  D4 000A7 5$:      CLRL    SKIP_MSG_DSC_ADDR                             ; 1087
                                          03  BB 000A9 6$:      PUSHR   #^M<R0,R1>                                    ; 1090
                          00000000G       EF  9F 000AB          PUSHAB  MOM$GQ_MOP_RCV_BUF_DSC                        ; 1088
                                EE        A5  9F 000B1          PUSHAB  MOM$AB_CIB
                                10        AE  9F 000B4          PUSHAB  SNDDSC
                                EE        A5  9F 000B7          PUSHAB  MOM$AB_CIB
                      00000000G EF              06  FB 000BA          CALLS   #6, MOM$MOPSNDRCV                         ; 1090
                          52              50  D0 000C1          MOVL    R0, STATUS
                          0D              52  E8 000C4          BLBS    STATUS, 7$                                    ; 1098
                          09      04      BC  E8 000C7          BLBS    @LOADFLAG, 7$
                                          52  DD 000CB          PUSHL   STATUS                                        ; 1099
                      00000000G EF              01  FB 000CD          CALLS   #1, MOM$CHK_MOP_ERROR
                          65              05  D0 000D4 7$:      MOVL    #5, MOM$AB_CIB+T8                             ; 1105
                          0C              52  E9 000D7          BLBC    STATUS, 9$                                    ; 1111
                    06    64              05  E0 000DA          BBS     #5, MOM$GL_SERVICE_FLAGS, 8$                  ; 1113
                          64      80      8F  8A 000DE          BICB2   #128, MOM$GL_SERVICE_FLAGS                    ; 1114
                                          02  11 000E2          BRB     9$
                                          52  D4 000E4 8$:      CLRL    STATUS                                        ; 1122
                          50              52  D0 000E6 9$:      MOVL    STATUS, R0                                    ; 1124
                                              04 000E9          RET                                                   ; 1126

; Routine Size:  234 bytes,    Routine Base:  $CODE$ + 0605

; 1135          1127  1
```

```
;  1137          1128  1 %SBTTL 'mom_openloadfile        Open the image file for loading'
;  1138          1129  1 ROUTINE mom_openloadfile =
;  1139          1130  1
;  1140          1131  1 !++
;  1141          1132  1 ! FUNCTIONAL DESCRIPTION:
;  1142          1133  1 !
;  1143          1134  1 !       Open the image file to be loaded and check the validity of the image.
;  1144          1135  1 !
;  1145          1136  1 ! FORMAL PARAMETERS:
;  1146          1137  1 !
;  1147          1138  1 !       NONE
;  1148          1139  1 !
;  1149          1140  1 !
;  1150          1141  1 ! IMPLICIT OUTPUTS:
;  1151          1142  1 !
;  1152          1143  1 !       MOM$W_PGMDETAIL Detail code to use for file errors.
;  1153          1144  1 !
;  1154          1145  1 ! ROUTINE VALUE:
;  1155          1146  1 ! COMPLETION CODES:
;  1156          1147  1 !
;  1157          1148  1 !       If no file name or service device is specified then FALSE is
;  1158          1149  1 !       returned indicating that not enough information was specified.
;  1159          1150  1 !       A FALSE return value indicates to the calling routine that
;  1160          1151  1 !       the target system must supply the missing information.  Any
;  1161          1152  1 !       errors encountered when trying to open the file will be signalled.
;  1162          1153  1 !
;  1163          1154  1 ! SIDE EFFECTS:
;  1164          1155  1 !
;  1165          1156  1 !       NONE
;  1166          1157  1 !--
;  1167          1158  1
;  1168          1159  2 BEGIN
;  1169          1160  2
;  1170          1161  2 LOCAL
;  1171          1162  2     adr,
;  1172          1163  2     dev,
;  1173          1164  2     fildsc      : VECTOR [2],
;  1174          1165  2     len,
;  1175          1166  2     msgsize,
;  1176          1167  2     file_svd_index,
;  1177          1168  2     ptr,
;  1178          1169  2     status;
;  1179          1170  2
;  1180          1171  2 !
;  1181          1172  2 ! Get the file type.
;  1182          1173  2 !
;  1183          1174  2 SELECTONEU .mom$ab_service_data [svd$gk_pcno_sty, svd$l_param] OF
;  1184          1175  2     SET
;  1185          1176  2     [nma$c_soft_terl]:          ! Tertiary loader
;  1186          1177  3         BEGIN
;  1187          1178  3         file_svd_index = svd$gk_pcno_tlo;
;  1188          1179  3         mom$w_pgmdetail = nma$c_fopdtl_tlf;
;  1189          1180  3         END;
;  1190          1181  2
;  1191          1182  2     [nma$c_soft_osys]:          ! Operating system or diagnostics
;  1192          1183  3         BEGIN
;  1193          1184  3         IF .mom$ab_service_data [svd$gk_pcno_$fty, svd$l_param] EQL
```

```
: 1194          1185  3                                         mop$c_sid_osy OR
: 1195          1186                  NOT .mom$gl_service_flags [mom$v_autoservice] THEN
: 1196          1187                      !
: 1197          1188                      ! Ignore requests for the diagnostics if the operator
: 1198          1189                      ! requested the load.  This is in case the test button
: 1199          1190                      ! has been left in on the target.
: 1200          1191                      !
: 1201          1192                      file_svd_index = svd$gk_pcno_loa
: 1202          1193                  ELSE
: 1203          1194                      file_svd_index = svd$gk_pcno_dfl;
: 1204          1195                  mom$w_pgmdetail = nma$c_fopdtl_lfl;
: 1205          1196                  END;
: 1206          1197
: 1207          1198              [OTHERWISE]:                        ! Secondary loader
: 1208          1199                  BEGIN
: 1209          1200                  file_svd_index = svd$gk_pcno_slo;
: 1210          1201                  mom$w_pgmdetail = nma$c_fopdtl_slf;
: 1211          1202                  END;
: 1212          1203
: 1213          1204              TES;
: 1214          1205      !
: 1215          1206      ! Get the file name of the file to be loaded.
: 1216          1207      !
: 1217          1208      IF .mom$ab_service_data [.file_svd_index, svd$b_string_len] EQL 0 THEN
: 1218          1209          BEGIN
: 1219          1210          !
: 1220          1211          ! File was not found in the data base so build it from the file type
: 1221          1212          ! and the service circuit.
: 1222          1213          !
: 1223          1214          ptr = mom$ab_service_data [.file_svd_index, svd$t_string];
: 1224          1215
: 1225          1216          SELECTONEU .mom$ab_service_data [svd$gk_pcno_sty, svd$l_param] OF
: 1226          1217              SET
: 1227          1218              [nma$c_soft_secl]:
: 1228          1219                  ptr = CA$MOVE (3, UPLIT BYTE ('SEC'), .ptr);
: 1229          1220
: 1230          1221              [nma$c_soft_terl]:
: 1231          1222                  ptr = CA$MOVE (3, UPLIT BYTE ('TER'), .ptr);
: 1232          1223
: 1233          1224              [OTHERWISE]:
: 1234          1225  4               BEGIN
: 1235          1226  4               mom$error (nma$c_sts_pms, nma$c_pcno_loa);
: 1236          1227  4               RETURN false;
: 1237          1228                  END;
: 1238          1229
: 1239          1230              TES;
: 1240          1231          !
: 1241          1232          ! Get the service device type code from the data base.
: 1242          1233          !
: 1243          1234          dev = .mom$ab_service_data [svd$gk_pcno_sdv, svd$l_param];
: 1244          1235          !
: 1245          1236          ! Get the service device name string from the table.
: 1246          1237          !
: 1247          1238          status = false;
: 1248          1239          INCR i FROM 0 TO mdt$gk_mopdevcnt - 1 DO
: 1249          1240  4           BEGIN
: 1250          1241  4           !
```

```
: 1251          1242  4          ! If a match is found in the table then move the device name string into
: 1252          1243  4          ! the file name buffer.
: 1253          1244  4          !
: 1254          1245  4          IF .mom$ab_mopdevices [.i, mdt$b_devtype] EQL .dev THEN
: 1255          1246  5              BEGIN
: 1256          1247  5              adr = .mom$ab_mopdevices [.i, mdt$a_devstring];
: 1257          1248  5              len = .(.adr)<0,8>;
: 1258          1249  5              adr = .adr + 1;
: 1259          1250  5              ptr = CH$MOVE (.len, .adr, .ptr);
: 1260          1251  5              status = true;
: 1261          1252  5              EXITLOOP;
: 1262          1253  4              END;
: 1263          1254  3          END;
: 1264          1255  3      !
: 1265          1256  3      ! If a service device was found in the table then set up the file name
: 1266          1257  3      ! descriptor.
: 1267          1258  3      !
: 1268          1259  3      IF .status THEN
: 1269          1260  3          mom$ab_service_data [.file_svd_index, svd$b_string_len] =
: 1270          1261  3                  .ptr - mom$ab_service_data [.file_svd_index, svd$t_string]
: 1271          1262  3      ELSE
: 1272          1263  4          BEGIN
: 1273          1264  4          mom$ab_msgblock [msb$l_flags] = msb$m_msg_fld;
: 1274          1265  4          mom$ab_msgblock [msb$l_text]  = mom$_unsmopdev;
: 1275          1266  4          mom$bld_reply (mom$ab_msgblock, msgsize);
: 1276          1267  4          $signal_msg (mom$ab_nice_xmit_buf, .msgsize);
: 1277          1268  3          END;
: 1278          1269  2      END;
: 1279          1270  2  fildsc [0] = .mom$ab_service_data [.file_svd_index, svd$b_string_len];
: 1280          1271  2  fildsc [1] = mom$ab_service_data [.file_svd_index, svd$t_string];
: 1281          1272  2  !
: 1282          1273  2  ! Open the file to be loaded.
: 1283          1274  2  !
: 1284          1275  2  status = mom$srvopen (fildsc, nma$c_opn_ac_ro);
: 1285          1276  2  !
: 1286          1277  2  ! If the file could not be opened then build and signal an error message.
: 1287          1278  2  !
: 1288          1279  2  IF NOT .status THEN
: 1289          1280  3      BEGIN
: 1290          1281  3      mom$ab_msgblock [msb$w_detail] = .mom$w_pgmdetail;
: 1291          1282  3      mom$bld_reply (mom$ab_msgblock, msgsize);
: 1292          1283  3      $signal_msg (mom$ab_nice_xmit_buf, .msgsize);
: 1293          1284  3      RETURN .status;
: 1294          1285  2      END;
: 1295          1286  2  !
: 1296          1287  2  ! Read in the first label block of the load file.  Get the load file attributes
: 1297          1288  2  ! from the block, and then skip over the rest of the label blocks to the
: 1298          1289  2  ! beginning of the load file image data.  Note the the console carrier system
: 1299          1290  2  ! load file skips this because it does not need, and therefore, does not have
: 1300          1291  2  ! a label block.
: 1301          1292  2  !
: 1302          1293  2  IF NOT .mom$gl_service_flags [mom$v_console_carrier_load] OR
: 1303          1294  2     .mom$ab_service_data [svd$gk_pcno_sty, svd$l_param] EQL nma$c_soft_secl THEN
: 1304          1295  2      mom_check_label_blk ();
: 1305          1296  2  RETURN true
: 1306          1297  2
: 1307          1298  1  END;                                 ! End of mom_openloadfile
```

```
                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                                        43  45  53  00114 P.AAS:  .ASCII  \SEC\                                                  ;
                                        52  45  54  00117 P.AAT:  .ASCII  \TER\                                                  ;


                                                        .PSECT  $CODE$,NOWRT,2

                                            0FFC 00000 MOM_OPENLOADFILE:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11                            ; 1129
                                    5E                  14  C2 00002          SUBL2   #20, SP
                                    51 00000000*        EF  D0 00005          MOVL    <<MOM$AB_SERVICE_DATA+<SVD$GK_PCNO_STY*137>-    ; 1174
                                                                                      >+9>, R1
                                    01                  51  D1 0000C          CMPL    R1, #1                                      ; 1176
                                                        10  12 0000F          BNEQ    1$
                                    50 00000000G        BF  D0 00011          MOVL    #SVD$GK_PCNO_TLO, FILE_SVD_INDEX            ; 1178
                        00000000'   EF                  04  B0 00018          MOVW    #4, MOM$W_PGMDETAIL                         ; 1179
                                                        40  11 0001F          BRB     6$                                          ; 1174
                                    02                  51  D1 00021 1$:      CMPL    R1, #2                                      ; 1182
                                                        2D  12 00024          BNEQ    5$
                        000000FF    8F 00000000*        EF  D1 00026          CMPL    <<MOM$AB_SERVICE_DATA+<SVD$GK_PCNO_$FTY*-        ; 1184
                                                                                      137>>+9>, #255
                                                        07  13 00031          BEQL    2$
                                    09 00000000G        EF  E8 00033          BLBS    MOM$GL_SERVICE_FLAGS, 3$                    ; 1186
                                    50 00000000G        8F  D0 0003A 2$:      MOVL    #SVD$GK_PCNO_LOA, FILE_SVD_INDEX            ; 1192
                                                        07  11 00041          BRB     4$
                                    50 00000000G        8F  D0 00043 3$:      MOVL    #SVD$GK_PCNO_DFL, FILE_SVD_INDEX            ; 1194
                            00000000'   EF              01  B0 0004A 4$:      MOVW    #1, MOM$W_PGMDETAIL                         ; 1195
                                                        0E  11 00051          BRB     6$                                          ; 1174
                                    50 00000000G        8F  D0 00053 5$:      MOVL    #SVD$GK_PCNO_SLO, FILE_SVD_INDEX            ; 1200
                            00000000'   EF              03  B0 0005A          MOVW    #3, MOM$W_PGMDETAIL                         ; 1201
                        57          50 00000089         8F  C5 00061 6$:      MULL3   #137, FILE_SVD_INDEX, R7                    ; 1208
                                    5A 00000000GEF47    9E  00069             MOVAB   MOM$AB_SERVICE_DATA+8[R7], R10
                                                        6A  95 00071          TSTB    (R10)
                                                        03  13 00073          BEQL    7$
                                                    00BD    31 00075          BRW     16$
                                    59 00000000GEF47    9E  00078 7$:      MOVAB   MOM$AB_SERVICE_DATA+9[R7], R9               ; 1214
                                    53                  59  D0 00080          MOVL    R9, PTR
                                                        51  D5 00083          TSTL    R1                                          ; 1218
                                                        0B  12 00085          BNEQ    8$
                    63      18          00 00000000'    EF  F0 00087          INSV    P.AAS, #0, #24, (PTR)                       ; 1219
                                                        0E  11 00090          BRB     9$
                                    01                  51  D1 00092 8$:      CMPL    R1, #1                                      ; 1221
                                                        0E  12 00095          BNEQ    10$
                    63      18          00 00000000'    EF  F0 00097          INSV    P.AAT, #0, #24, (PTR)                       ; 1222
                                    53                  03  C0 000A0 9$:      ADDL2   #3, PTR
                                                        11  11 000A3          BRB     11$
                                    7E          78      8F  9A 000A5 10$:     MOVZBL  #120, -(SP)                                 ; 1226
                                    7E                  1D  CE 000A9          MNEGL   #29, -(SP)
                            00000000G   EF              02  FB 000AC          CALLS   #2, MOM$ERROR
                                                    00EE    31 000B3          BRW     20$                                         ; 1227
                                04      AE 00000000*    EF  D0 000B6 11$:     MOVL    <<MOM$AB_SERVICE_DATA+<SVD$GK_PCNO_SDV*137>-    ; 1234
                                                                                      >+9>, DEV
```

```
                                                6E  D4 000BE               CLRL    STATUS                                          ; 1238
                                             56 01  CE 000C0               MNEGL   #1, I                                           ; 1245
                                                27  11 000C3               BRB     13$
                                          50 56 05  C5 000C5 12$:          MULL3   #5, I, R0
                  04   AE 00000000GEF40    08 00  ED 000C9               CMPZV   #0, #8, MOM$AB_MOPDEVICES[R0], DEV
                                                16  12 000D4               BNEQ    13$
                                   00000000GEF40    9F 000D6               PUSHAB  MOM$AB_MOPDEVICES+1[R0]                         ; 1247
                                             58 9E  D0 000DD               MOVL    @(SP)+, ADR
                                             5B 88  9A 000E0               MOVZBL  (ADR)+, LEN                                     ; 1248
                                       63    68 5B  28 000E3               MOVC3   LEN, (ADR), (PTR)                               ; 1250
                                             6E 01  D0 000E7               MOVL    #1, STATUS                                      ; 1251
                                                08  11 000EA               BRB     14$                                             ; 1246
                                       D1    56 00000000G   8F  F3 000EC 13$:  AOBLEQ  #MDT$GK_MOPDEVCNT-1, I, 12$              ; 1239
                                             06 6E  E9 000F4 14$:             BLBC    STATUS, 15$                                     ; 1259
                                       6A    53 59  83 000F7               SUBB3   R9, PTR, (R10)                                  ; 1261
                                                38  11 000FB               BRB     16$                                             ; 1260
                                   00000000G   EF 04  D0 000FD 15$:          MOVL    #4, MOM$AB_MSGBLOCK                             ; 1264
                                   00000000G   EF 00000000G   8F  D0 00104   MOVL    #MOM$_UNSMOPDEV, MOM$AB_MSGBLOCK+12             ; 1265
                                                   08  AE 9F 0010F         PUSHAB  MSGSIZE                                         ; 1266
                                          00000000G    EF 9F 00112         PUSHAB  MOM$AB_MSGBLOCK
                                   00000000G   EF 02  FB 00118               CALLS   #2, MOM$BLD_REPLY
                                                   08  AE DD 0011F         PUSHL   MSGSIZE                                         ; 1267
                                          00000000G    EF 9F 00122         PUSHAB  MOM$AB_NICE_XMIT_BUF
                                          02070000     8F DD 00128         PUSHL   #34013784
                                   00000000G   00 03  FB 0012E               CALLS   #3, LIB$SIGNAL
                                         0C    AE 6A  9A 00135 16$:          MOVZBL  (R10), FILDSC                                   ; 1270
                                         10    AE 00000000GEF47  9E 00139    MOVAB   MOM$AB_SERVICE_DATA+9[R7], FILDSC+4             ; 1271
                                                7E  D4 00142               CLRL    -(SP)                                           ; 1275
                                                   10  AE 9F 00144         PUSHAB  FILDSC
                                   00000000G   EF 02  FB 00147               CALLS   #2, MOM$SRVOPEN
                                             6E 50  D0 0014E               MOVL    R0, STATUS
                                             35 6E  E8 00151               BLBS    STATUS, 17$                                     ; 1279
                                   00000000G   EF 00000000'   EF  B0 00154   MOVW    MOM$W_PGMDETAIL, MOM$AB_MSGBLOCK+8              ; 1281
                                                   08  AE 9F 0015F         PUSHAB  MSGSIZE                                         ; 1282
                                          00000000G    EF 9F 00162         PUSHAB  MOM$AB_MSGBLOCK
                                   00000000G   EF 02  FB 00168               CALLS   #2, MOM$BLD_REPLY
                                                   08  AE DD 0016F         PUSHL   MSGSIZE                                         ; 1283
                                          00000000G    EF 9F 00172         PUSHAB  MOM$AB_NICE_XMIT_BUF
                                          02070000     8F DD 00178         PUSHL   #34013784
                                   00000000G   00 03  FB 0017E               CALLS   #3, LIB$SIGNAL
                                             50 6E  D0 00185               MOVL    STATUS, R0                                      ; 1284
                                                04  00188               RET
                                08 00000000G   EF 06  E1 00189 17$:          BBC     #6, MOM$GL_SERVICE_FLAGS, 18$                   ; 1293
                                          00000000*    EF D5 00191         TSTL    <<MOM$AB_SERVICE_DATA+<SVD$GK_PCNO_STY*137>-    ; 1294
                                                                                    >+9>
                                                07  12 00197               BNEQ    19$
                                   00000000V   EF 00  FB 00199 18$:          CALLS   #0, MOM_CHECK_LABEL_BLK                         ; 1295
                                             50 01  D0 001A0 19$:          MOVL    #1, R0                                          ; 1296
                                                04  001A3               RET
                                                50  D4 001A4 20$:          CLRL    R0                                              ; 1298
                                                04  001A6               RET

; Routine Size:  423 bytes,    Routine Base:  $CODE$ + 06EF

; 1308          1299  1
```

```
;  1310          1300  1 %SBTTL 'mom_readloadfile       Read a block from the image file'
;  1311          1301  1 ROUTINE mom_readloadfile (read_buf_dsc) : NOVALUE =
;  1312          1302  1
;  1313          1303  1 !++
;  1314          1304  1 ! FUNCTIONAL DESCRIPTION:
;  1315          1305  1 !
;  1316          1306  1 !       This routine reads a block from the load file that is currently open.
;  1317          1307  1 !
;  1318          1308  1 ! FORMAL PARAMETERS:
;  1319          1309  1 !
;  1320          1310  1 !       READ_BUF_DSC    Address of read buffer descriptor.
;  1321          1311  1 !
;  1322          1312  1 ! IMPLICIT INPUTS:
;  1323          1313  1 !
;  1324          1314  1 !       The load file to be read is open.
;  1325          1315  1 !
;  1326          1316  1 !       MOM$L_LOADSIZE  .
;  1327          1317  1 !       MOM$W_PGMDETAIL .
;  1328          1318  1 !
;  1329          1319  1 ! IMPLICIT OUTPUTS:
;  1330          1320  1 !
;  1331          1321  1 !       MOM$Q_DATADSC   Describes the extent of the data that was read.
;  1332          1322  1 !       MOM$L_BLKCNT    Number of 64-byte blocks in the buffer.
;  1333          1323  1 !
;  1334          1324  1 ! ROUTINE VALUE:
;  1335          1325  1 ! COMPLETION CODES:
;  1336          1326  1 !
;  1337          1327  1 !       Signal errors.
;  1338          1328  1 !
;  1339          1329  1 ! SIDE EFFECTS:
;  1340          1330  1 !
;  1341          1331  1 !       NONE
;  1342          1332  1 !--
;  1343          1333  1
;  1344          1334  2 BEGIN
;  1345          1335  2
;  1346          1336  2 LOCAL
;  1347          1337  2     msgsize,
;  1348          1338  2     status;
;  1349          1339  2
;  1350          1340  2 !
;  1351          1341  2 ! Read as many records from the load file as will fit into the read buffer
;  1352          1342  2 ! and return the byte count of the data read in MOM$Q_DATADSC.
;  1353          1343  2 !
;  1354          1344  2 mom$srvread (.read_buf_dsc,
;  1355          1345  2               mom$q_datadsc [0],
;  1356          1346  2               .mom$w_pgmdetail);
;  1357          1347  2 !
;  1358          1348  2 ! Return the number of 64 byte blocks.  If the number of blocks in the
;  1359          1349  2 ! buffer is less than the number of blocks remaining to be loaded then
;  1360          1350  2 ! use the number to be loaded.  This will account for extra blocks that
;  1361          1351  2 ! were the result of zero-filling.
;  1362          1352  2 !
;  1363          1353  2 mom$l_blkcnt = .mom$q_datadsc [0] / 64;
;  1364          1354  2 !
;  1365          1355  2 ! If the number of blocks is zero then the byte count of the record that was
;  1366          1356  2 ! read was not valid.  The record size must be a multiple of 64.
```

```
;  1367          1357  2 !
;  1368          1358  2 IF .mom$l_blkcnt EQLU 0 THEN
;  1369          1359  2     BEGIN
;  1370          1360  2
;  1371          1361  2     mom$ab_msgblock [msb$l_flags]  = msb$m_msg_fld;
;  1372          1362  2     mom$ab_msgblock [msb$b_code]   = nma$c_sts_fco;
;  1373          1363  2     mom$ab_msgblock [msb$w_detail] = .mom$w_pgmdetail;
;  1374          1364  2     mom$ab_msgblock [msb$l_text]   = mom$_imgrecsiz;
;  1375          1365  2     mom$bld_reply (mom$ab_msgblock, msgsize);
;  1376          1366  2     $signal_msg (mom$ab_nice_xmit_buf, .msgsize);
;  1377          1367  2
;  1378          1368  2     END;
;  1379          1369  2
;  1380          1370  2 IF .mom$l_blkcnt GTRU .mom$l_loadsize THEN
;  1381          1371  2     mom$l_blkcnt = .mom$l_loadsize;
;  1382          1372  2
;  1383          1373  1 END;                                ! End of mom_readloadfile



                                        000C 00000 MOM_READLOADFILE:
                                                            .WORD   Save R2,R3                                      ; 1301
                                53 00000000G    EF  9E 00002        MOVAB   MOM$AB_MSGBLOCK, R3                     ;
                                52 00000000'    EF  9E 00009        MOVAB   MOM$L_BLKCNT, R2                        ;
                                5E              04  C2 00010        SUBL2   #4, SP                                  ;
                                7E        0C    A2  3C 00013        MOVZWL  MOM$W_PGMDETAIL, -(SP)                  ; 1346
                                        0C20    C2  9F 00017        PUSHAB  MOM$Q_DATADSC                           ; 1345
                                          04    AC  DD 0001B        PUSHL   READ_BUF_DSC                            ; 1344
                                00000000G       EF  03 FB 0001E     CALLS   #3, MOM$$RVREAD                         ;
                             62   0C20          C2  00000040   8F  C7 00025        DIVL3   #64, MOM$Q_DATADSC, MOM$L_BLKCNT        ; 1353
                                                        34  12 0002F        BNEQ    1$                              ; 1358
                                          63                04  D0 00031        MOVL    #4, MOM$AB_MSGBLOCK             ; 1361
                                    04    A3                0E  8E 00034        MNEGB   #14, MOM$AB_MSGBLOCK+4          ; 1362
                                    08    A3          0C    A2  B0 00038        MOVW    MOM$W_PGMDETAIL, MOM$AB_MSGBLOCK+8      ; 1363
                                    0C    A3    00000000G   8F  D0 0003D        MOVL    #MOM$_IMGRECSIZ, MOM$AB_MSGBLOCK+12     ; 1364
                                                  4008      8F  BB 00045        PUSHR   #^M<R3,SP>                      ; 1365
                                00000000G       EF              02  FB 00049    CALLS   #2, MOM$BLD_REPLY               ;
                                                                6E  DD 00050    PUSHL   MSGSIZE                         ; 1366
                                          00000000G             EF  9F 00052    PUSHAB  MOM$AB_NICE_XMIT_BUF            ;
                                          02070000              8F  DD 00058    PUSHL   #34013784                       ;
                                00000000G 00                    03  FB 0005E    CALLS   #3, LIB$SIGNAL                  ;
                                    04    A2                    62  D1 00065 1$: CMPL    MOM$L_BLKCNT, MOM$L_LOADSIZE    ; 1370
                                                                04  1B 00069    BLEQU   2$                              ;
                                          62            04    A2  D0 00068    MOVL    MOM$L_LOADSIZE, MOM$L_BLKCNT    ; 1371
                                                                04  0006F 2$:   RET                                     ; 1373

; Routine Size:  112 bytes,    Routine Base:  $CODE$ + 0896

; 1384          1374  1
```

```
;  1386          1375  1 %SBTTL 'mom_check_label_blk       Perform file label block check'
;  1387          1376  1 ROUTINE mom_check_label_blk : NOVALUE =
;  1388          1377  1
;  1389          1378  1 !++
;  1390          1379  1 ! FUNCTIONAL DESCRIPTION:
;  1391          1380  1 !
;  1392          1381  1 !       The load files are assumed to be built by the RSX11M task image
;  1393          1382  1 !       builder.  Read in the file label blocks and extract the information
;  1394          1383  1 !       required to down line load the image in the file.
;  1395          1384  1 !
;  1396          1385  1 ! IMPLICIT INPUTS:
;  1397          1386  1 !
;  1398          1387  1 !       MOM$W_PGMDETAIL Detail code to use for file contents errors.
;  1399          1388  1 !
;  1400          1389  1 ! IMPLICIT OUTPUTS:
;  1401          1390  1 !
;  1402          1391  1 !       MOM$L_LOADSIZE = the size of the image to be down line loaded.
;  1403          1392  1 !                        The size is specified in number of 32 word blocks.
;  1404          1393  1 !       MOM$L_BASEADR =  The address at which to start loading the image
;  1405          1394  1 !                        into the target node's memory.
;  1406          1395  1 !       MOM$L_TRANSFER = The address at which to start executing the image
;  1407          1396  1 !                        once it has been down line loaded to the target node.
;  1408          1397  1 !
;  1409          1398  1 ! ROUTINE VALUE:
;  1410          1399  1 ! COMPLETION CODES:
;  1411          1400  1 !
;  1412          1401  1 !       Signal errors.
;  1413          1402  1 !
;  1414          1403  1 !--
;  1415          1404  1
;  1416          1405  2 BEGIN
;  1417          1406  2
;  1418          1407  2 !
;  1419          1408  2 ! Define RSX label block symbols.
;  1420          1409  2 !
;  1421          1410  2 EXTERNAL LITERAL
;  1422          1411  2     l$bflg,                 ! Word
;  1423          1412  2     l$bhgv,                 ! Word
;  1424          1413  2     l$bmxv,                 ! Word
;  1425          1414  2     l$bldz,                 ! Word
;  1426          1415  2     l$bmxz,                 ! Word
;  1427          1416  2     l$bwnd,                 ! Byte
;  1428          1417  2     l$blib,                 ! Word
;  1429          1418  2     l$bsgl,                 ! Word
;  1430          1419  2     l$boff,                 ! Word
;  1431          1420  2     l$bblk,                 ! Word
;  1432          1421  2     l$bsa,                  ! Word
;  1433          1422  2     l$bxfr,                 ! Word
;  1434          1423  2     ts$nhd,
;  1435          1424  2     ts$chk,
;  1436          1425  2     ts$res;
;  1437          1426  2
;  1438          1427  2 LOCAL
;  1439          1428  2     label_buf_dsc : VECTOR [2], ! Descriptor for label read buffer
;  1440          1429  2     lbl : REF BBLOCK,           ! Pointer to label buffer
;  1441          1430  2     isd : REF BBLOCK,           ! Pointer to VMS image section desc
;  1442          1431  2     iha : REF BBLOCK;           ! Pointer to VMS image activation desc
```

```
;  1443          1432  2
;  1444          1433  2 label_buf_dsc [0] = 512;
;  1445          1434  2 label_buf_dsc [1] = mom$t_readbuffer;
;  1446          1435  2 !
;  1447          1436  2 ! Read the file label block.
;  1448          1437  2 !
;  1449          1438  2 mom_readloadfile (label_buf_dsc);
;  1450          1439  2
;  1451          1440  2 lbl = mom$t_readbuffer;
;  1452          1441  2 !
;  1453          1442  2 ! Determine whether image is an RSX-11 or VMS image.  This is done by
;  1454          1443  2 ! testing the last word in the image header.
;  1455          1444  2 !
;  1456          1445  2 IF .lbl[510,0,16,1] GEQ 0 THEN
;  1457          1446  2     BEGIN
;  1458          1447  2     !
;  1459          1448  2     ! Save the RSX task image information from the label block.
;  1460          1449  2     !
;  1461          1450  2     mom$l_blkcnt = .lbl [l$bblk,0,16,0];
;  1462          1451  2     mom$l_loadsize = .lbl [l$bldz,0,16,0]; ! Image size (32-word blocks)
;  1463          1452  2     mom$l_baseadr  = .lbl [l$bsa ,0,16,0]; ! Starting memory address
;  1464          1453  2     mom$l_transfer = .lbl [l$bxfr,0,16,0]; ! Image transfer address
;  1465          1454  2
;  1466          1455  2     END
;  1467          1456  2 ELSE
;  1468          1457  2     BEGIN
;  1469          1458  2     !
;  1470          1459  2     ! Save the VMS image information from the image header block.
;  1471          1460  2     !
;  1472          1461  3     mom$l_blkcnt = .lbl [ihd$b_hdrblkcnt];
;  1473          1462  3     isd = .lbl [ihd$w_size] + .lbl;              ! Get first image section desc
;  1474          1463  3     mom$l_loadsize = .isd [isd$w_pagcnt] * 8;    ! Image size (32-word blocks)
;  1475          1464  3     mom$l_baseadr  = .isd [isd$v_vpn] * 512;     ! Starting memory address
;  1476          1465  3     iha = .lbl [ihd$w_activoff] + .lbl;
;  1477          1466  3     mom$l_transfer = .iha [iha$l_tfradr1];       ! Image transfer address
;  1478          1467  3     mom$l_transfer = .mom$l_transfer<0,31,0>;    ! (remove S0 bit)
;  1479          1468  3
;  1480          1469  2     END;
;  1481          1470  2 !
;  1482          1471  2 ! Read past the load file label blocks to the beginning of the image to be
;  1483          1472  2 ! loaded.  (The first label block has already been read.)
;  1484          1473  2 !
;  1485          1474  2 DECR i FROM .mom$l_blkcnt - 2 DO
;  1486          1475  3     BEGIN
;  1487          1476  3     mom_readloadfile (label_buf_dsc);
;  1488          1477  2     END;
;  1489          1478  2 RETURN
;  1490          1479  2
;  1491          1480  1 END;                                 ! End of mom_check_label_blk


                                                                .EXTRN  L$BFLG, L$BHGV, L$BMXV
                                                                .EXTRN  L$BLDZ, L$BMXZ, L$BWND
                                                                .EXTRN  L$BLIB, L$BSGL, L$BOFF
                                                                .EXTRN  L$BBLK, L$BSA, L$BXFR
                                                                .EXTRN  TS$NHD, TS$CHK, TS$RES
```

```
                                                M 16
                                001C 00000 MOM_CHECK_LABEL_BLK:
                                            .WORD   Save R2,R3,R4                                                   ; 1376
                          54        8B    AF  9E 00002          MOVAB   MOM_READLOADFILE, R4                          ;
                          53 00000000'    EF  9E 00006          MOVAB   MOM$L_TRANSFER, R3                            ;
                          5E              04  C2 0000D          SUBL2   #4, SP                                        ;
                          7E        0200  8F  3C 00010          MOVZWL  #512, LABEL_BUF_DSC                           ; 1433
                       04 AE        0612  C3  9E 00015          MOVAB   MOM$T_READBUFFER, LABEL_BUF_DSC+4             ; 1434
                                          5E  DD 00019          PUSHL   SP                                            ; 1438
                          64              01  FB 0001D          CALLS   #1, MOM_READLOADFILE                          ;
                          50        0612  C3  9E 00020          MOVAB   MOM$T_READBUFFER, LBL                         ; 1440
                                    01FE  C0  B5 00025          TSTW    510(LBL)                                      ; 1445
                                          21  19 00029          BLSS    1$                                            ;
                       F8 A3 00000000G    E0  3C 0002B          MOVZWL  L$BBLK(LBL), MOM$L_BLKCNT                     ; 1450
                       FC A3 00000000G    E0  3C 00033          MOVZWL  L$BLDZ(LBL), MOM$L_LOADSIZE                   ; 1451
                       F4 A3 00000000G    E0  3C 0003B          MOVZWL  L$BSA(LBL), MOM$L_BASEADR                     ; 1452
                          63 00000000G    E0  3C 00043          MOVZWL  L$BXFR(LBL), MOM$L_TRANSFER                   ; 1453
                                          2E  11 0004A          BRB     2$                                            ; 1445
                       F8 A3        10    A0  9A 0004C 1$:      MOVZBL  16(LBL), MOM$L_BLKCNT                         ; 1461
                          51              60  3C 00051          MOVZWL  (LBL), ISD                                    ; 1462
                          51              50  C0 00054          ADDL2   LBL, ISD                                      ;
                          52        02    A1  3C 00057          MOVZWL  2(ISD), R2                                    ; 1463
                    FC A3                 52  03 78 0005B          ASHL    #3, R2, MOM$L_LOADSIZE                        ;
                 52 04 A1                 15  00 EF 00060          EXTZV   #0, #21, 4(ISD), R2                           ; 1464
                    F4 A3                 52  09 78 00066          ASHL    #9, R2, MOM$L_BASEADR                         ;
                          51        02    A0  3C 0006B          MOVZWL  2(LBL), R1                                    ; 1465
                          50              51  C0 0006F          ADDL2   R1, IHA                                       ;
                          63              60  D0 00072          MOVL    (IHA), MOM$L_TRANSFER                         ; 1466
                 63             63        1F  00 EF 00075          EXTZV   #0, #31, MOM$L_TRANSFER, MOM$L_TRANSFER       ; 1467
                          52 F8 A3              01  C3 0007A 2$:      SUBL3   #1, MOM$L_BLKCNT, I                           ; 1474
                                                05  11 0007F          BRB     4$                                            ;
                                                5E  DD 00081 3$:      PUSHL   SP                                            ; 1476
                                             64     01  FB 00083          CALLS   #1, MOM_READLOADFILE                          ;
                                             F8     52  F4 00086 4$:      SOBGEQ  I, 3$                                         ; 1474
                                                    04 00089          RET                                                   ; 1480

; Routine Size:  138 bytes,    Routine Base:  $CODE$ + 0906
```

```
;  1493          1481  1 %SBTTL 'mom$loadhandler  Condition handler'
;  1494          1482  1 GLOBAL ROUTINE mom$loadhandler (signal_vec, mechanism)  =
;  1495          1483  1
;  1496          1484  1 !++
;  1497          1485  1 ! FUNCTIONAL DESCRIPTION:
;  1498          1486  1 !
;  1499          1487  1 !       This routine is a condition handler that performs cleanup
;  1500          1488  1 !       at the end of load operations.  All files are closed.
;  1501          1489  1 !
;  1502          1490  1 ! FORMAL PARAMETERS:
;  1503          1491  1 !
;  1504          1492  1 !       SIGNAL_VEC      Pointer to the signal vector.
;  1505          1493  1 !       MECHANISM       Pointer to the mechanism array.
;  1506          1494  1 !
;  1507          1495  1 !--
;  1508          1496  2 BEGIN
;  1509          1497  2
;  1510          1498  2 MAP
;  1511          1499  2     signal_vec : REF BBLOCK,    ! Signal vector argument
;  1512          1500  2     mechanism  : REF BBLOCK;    ! Mechanism vector array pointer
;  1513          1501  2
;  1514          1502  2 !
;  1515          1503  2 ! Close any open file.
;  1516          1504  2 !
;  1517          1505  2 mom$srvclose ();
;  1518          1506  2
;  1519          1507  2 RETURN ss$_resignal;            ! Always resignal error
;  1520          1508  2
;  1521          1509  1 END;                            ! End of mom$loadhandler


                                                0000 00000      .ENTRY  MOM$LOADHANDLER, Save nothing           ; 1482
                        00000000G   EF          00   FB 00002   CALLS   #0, MOM$SRVCLOSE                        ; 1505
                                    50      0918    8F   3C 00009   MOVZWL  #2328, R0                           ; 1507
                                                    04 0000E    RET                                             ; 1509

; Routine Size:  15 bytes,    Routine Base:  $CODE$ + 0990

;  1522          1510  1
;  1523          1511  1
;  1524          1512  1
;  1525          1513  1 END
;  1526          1514  1
;  1527          1515  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL

;                                   PSECT SUMMARY
;
;       Name                    Bytes                       Attributes
```

```
;
;  $OWN$                    3120  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                    282  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                   2463  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

```
;                          Library Statistics
;
;                                    --------- Symbols ---------      Pages       Processing
;       File                         Total     Loaded     Percent     Mapped      Time
;
;  _$255$DUA28:[MOM.OBJ]MOMLIB.L32;1      194         39          20          21        00:00.1
;  _$255$DUA28:[SHRLIB]NMALIBRY.L32;1     887         18           2          47        00:00.2
;  _$255$DUA28:[SHRLIB]EVCDEF.L32;1       213          2           0          15        00:00.1
;  _$255$DUA28:[SHRLIB]NET.L32;1         1279          0           0          63        00:00.4
;  _$255$DUA28:[SYSLIB]LIB.L32;1        18619         10           0        1000        00:06.6
```

```
;                                  COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:MOMLOAD/OBJ=OBJ$:MOMLOAD MSRC$:MOMLOAD/UPDATE=(ENH$:MOMLOAD)

; Size:          2463 code + 3402 data bytes
; Run Time:         00:46.0
; Elapsed Time:     01:28.3
; Lines/CPU Min:     1976
; Lexemes/CPU-Min: 11171
; Memory Used:  192 pages
; Compilation Complete
```

MOMBLDMSG
LIS

MOMINISTA
LIS

MOMLOAD
LIS

MOMACPIO
LIS

MOMDAT
LIS

NMAHEAD
B32

NPADEF
SDL

MOMDEF
SDL

NMATAIL
B32

MOMDDL
B32

NPAMAC
MAR

MOMLIB
LIS

MOMFILTO
LIS

MOMMSG
LIS

MOMMAIN
LIS

MOMPARSE
LIS

MOMSERSTA
LIS

MOMMOPSTA
LIS

MOMSERVIC
LIS

MOMMOPLIO
LIS

MOMTESSTA
LIS

MOMRSXDEF
LIS

MOMSUBS
LIS